Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jan Gulič

Ogrodje za navidezno resničnost z uporabo mobilnih naprav in senzorja Kinect

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Matija Marolt

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Zahvaljujem se mentorju doc. dr. Matiji Maroltu za potrpežljivost in usmerjanje pri izdelavi diplomske naloge. Iskrena hvala tudi as. dr. Cirilu Bohaku za pomoč in podporo pri izdelavi diplomske naloge.

Kazalo

Slike

Seznam uporabljenih kratic

kratica	angleško	slovensko
VR	virtual reality	Navidezna resničnost
OS	Operating system	Operacijski sistem
HMD	Head mounted display	Naglavni prikazovalnik
UDP	User Datagram Protocol	Nepovezovalni protokol za prenašanje paketov
3D	Three-dimensional	Tridimenzionalno
RGB	Red, Green, Blue	Barvni model, sestavljen iz rdeče, zelene in modre barve
GPU	Graphics processing unit	Grafična procesna enota
CAREN	Computer Assisted Rehabilitation Environment	Računalniško podprto rehabilitacijsko okolje
MS	Multiple sclerosis	Multipla skleroza
PTSD	Posttraumatic stress disorder	Posttravmatska stresna motnja
IR	Infrared	Infrardeče valovanje
CMOS	Complementary Metal Oxide Semiconductor	Komplementarni kovinsko-oksidni polprevodnik
FoV	Field of view	Vidno polje
ToF	Time of flight	Čas potovanja svetlobe
DoF	Degrees of freedom	Stopinje svobode gibanja
OLED	Organic Light Emitting Diode	Organska svetleča dioda
USB	Universal Serial Bus	Univerzalno serijsko vodilo
IMU	Inertial Measurement Unit	Inercialna merilna enota

PC	Personal computer	Osebni računalnik
SDK	Software development kit	Komplet programskih orodij za razvijanje programske opreme
FCL	Framework Class Library	Knjižnica razredov okolja
CLR	Common Language Runtime	Izvajalec kode za skupni jezik
IDE	Integrated development environment	Integrirano razvojno okolje
TCP	Transmission Control Protocol	Protokol za nadzor prenosa
UI	User interface	Uporabniški vmesnik
IP	Internet protocol	Internetni protokol

Povzetek

Naslov: Ogrodje za navidezno resničnost z uporabo mobilnih naprav in senzorja Kinect

V delu opisujemo razvoj aplikacije, ki združuje principe zaznavanja gibanja z navidezno resničnostjo. Zaznavanje gibanja je potekalo s pomočjo Microsoftovega senzorja Kinect, za doseganje izkušnje navidezne resničnosti pa smo uporabili Googlov Cardboard. V ta namen smo vzpostavili sistem, ki je obdelal in pretvoril Kinectove podatke v obliko, ki je bila primerna za delovanje v okolju Unity. Te je nato s pomočjo serializacije in povezave UDP pošiljal aplikaciji. Aplikacijo smo na koncu razvili v dve ločeni igri. Prva je namenjena enemu samemu igralcu, ki v navidezni resničnosti hodi nad prepadom. Druga igra se imenuje Rokoborba in je namenjena dvema igralcema, ki sta postavljena v boksarski ring, kjer si zadajata navidezne udarce. Za boljše razumevanje problema so bila preučena teoretična ozadja delovanja izbranih tehnologij. V prvem delu diplomske naloge je opisan pregled področij Kinecta in navidezne resničnosti, zatem pa sledi razlaga orodij in okolij, ki smo jih potrebovali za razvoj aplikacije. V drugem delu diplomske naloge smo opisali razvoj aplikacije. Podrobneje je razložen vsak korak razvoja, opisano je, kako aplikacija deluje, naštete so možne izboljšave programa ter predstavljeni so rezultati testiranja aplikacij.

Ključne besede: Senzor Kinect, navidezna resničnost, Cardboard, Unity, zaznavanje gibanja, aplikacija, razvoj, povezava UDP, strežnik - odjemalec, Microsoft Kinect SDK, Rokoborba, Hoja nad prepadom.

Abstract

Title: A virtual reality framework based on mobile devices and Kinect senzors

In this thesis we describe the development of an application that combines the principles of motion detection with virtual reality. Motion detection was carried out with the help of Microsoft's Kinect and we used Google's Cardboard to achieve the virtual reality experience. To this end, we have set up a system that has processed and converted Kinect data into a form that was suitable for operation in the Unity environment. We ultimately developed the application in two separate games. The first one is intended for a single player who, in virtual reality, is walking above the abyss. The second game is called "Rokoborba" and is intended for two players who are placed in a boxing ring, where they try to fight each other. To better understand the problem, we studied the theoretical background of the selected technologies. The first part of the thesis describes the Kinect and virtual reality technologies, followed by an explanation of the tools and environments that we needed to develop the application. In the second part of the thesis, we describe the development of the application. Each step of development is explained in more detail. We describe how does the application work and list possible improvements to the program. At the end we present the results of testing the applicatons.

Keywords: Kinect, virtual reality, Cardboard, Unity, motion detection, application, development, UDP connection, server - client, Microsoft Kinect SDK, Rokoborba, Hoja nad prepadom.

Poglavje 1

Uvod

Zaznavanje gibanja in navidezna resničnost sta pojma, ki osnujeta to diplomsko nalogo.

Zaznavanje gibanja je proces zaznavanja spremembe položaja predmeta glede na okolico oziroma spremembe položaja okolice glede na predmet. Ta je značilen tako za naprave kot tudi za organizme, ampak bomo zaradi narave diplomske naloge govorili le o prvih. Te naprave, ki s svojimi detektorji gibanja zaznavajo okolico, predmete in tudi nas - ljudi, so postale nenadomestljiv del našega sveta in imajo pomembno vlogo na veliko področjih. Uporabljajo se v medicini, transportu, v vojaške namene, multimediji in nasploh v vsakdanjem življenju. Z ultrazvokom odkrivamo bolezni pacienta ali pa preverimo stanje dojenčka v materinem trebuhu, imamo avtomobile, ki lahko s pomočjo različnih senzorjev nadomestijo voznika, luči, ki se prižgejo ob prisotnosti človeka, poznamo naprave, ki merijo seizmične valove in nas opozarjajo pred potresi, in tako dalje. Gibanje lahko zaznamo preko infrardečega sevanja, optike, radijskih valov, zvoka, vibracij in magnetizma [31].

Kaj pa je navidezna resničnost? Svet spoznavamo skozi svoje čute in zaznavne sisteme, kot so vid, sluh, dotik ... Vse kar vemo o svoji resničnosti je ustvarjeno s pomočjo njih. Z drugimi besedami, naša celotna podoba realnosti je le kombinacija senzoričnih informacij in čutnih odločitev naših

možganskih mehanizmov nanj [67]. Sledi torej, da če svojim čutom predstavimo 'izmišljene' informacije, bi naše dojemanje realnosti na to odgovorilo. Pred seboj bi imeli svet, ki ga pravzaprav ni, ampak bi se nam ta še vedno zdel resničen. Zakaj pa sploh želja po pobegu iz resničnosti v navidezni svet? Odgovor je preprost: Realen svet je en in edini, navideznih svetov je neskončno, te pa je mogoče oblikovati natanko tako, kot si jih želimo, za ustvarjanje izkušenj in situacij, ki niso drugače možne. Najbolj je navidezna resničnost prodrla na področje multimedije, medicine in letalstva. V medicini se uporablja kot pripomoček za usposabljanje in omogoča kirurgu, da opravi operacijo na 'navideznem pacientu' ter da vidi znotraj človeškega telesa. Uporablja se tudi kot diagnostično orodje, saj omogoča bolj podroben pogled na človeško telo v primerjavi z rentgenskimi žarki in pregledi. V letalstvu se navidezna resničnost uporablja za simulacije letenja. Te služijo za usposabljanje pilotov, za načrtovanje in razvoj letal ter za raziskovanje vpliva zunanjih dejavnikov na sam let. Navidezna resničnost pa ni aktualna samo na teh področjih, vedno večjo vlogo ima tudi v arhitekturi, poslovanju, športu, umetnosti, izobraževanju in še na mnogo drugih področjih. Ne glede na uporabo pa navidezna resničnost proizvaja niz podatkov, ki se nato uporabljajo za razvijanje novih modelov, novih metod usposabljanja, komunikacije in interakcije. Možnosti razvoja VR so zaradi tega tako rekoč neskončne.

Cilj naše diplomske je tako ustvariti ogrodje, ki bo temeljilo na prej omenjenih tehnologijah, in s pomočjo tega razviti dve aplikaciji. Uporabljale bodo podatke o gibanju iz realnega sveta in te spojili z navidezno resničnostjo (angl. Virtual Reality - VR). Za dosego ciljev bomo uporabili napravi Microsoft Kinect ter Google Cardboard. Kinect je naprava, namenjena zaznavanju gibanja in govora. To ji omogočajo različni senzorji, kot so globinski senzor, kamera ter mikrofoni. Sprva je bila uporabljena kot igralni pripomoček konzoli XBox, zdaj pa se uporablja tudi za reševanje drugih problemov. Googlov Cardboard pa je kartonasta naprava, škatlaste oblike, namenjena doživetju izkušnje navidezne resničnosti.

Zakaj uporabiti ravno ti dve napravi? Zakaj ne uporabiti alternativ?

Pripomočkov za zaznavanje človeškega gibanja in prikazovanja navidezne resničnosti je veliko in nekateri od teh ponujajo veliko več kot Kinect ter Cardboard. Zakaj potem ustvariti svoj sistem? Razlogov je več: Zaradi nizkih cen Kinecta in Cardboarda bo naš sistem bistveno cenejši za postavitev od ostalih. Kinect je v uporabi že precej let, zanj pa je bilo ustvarjenih že veliko knjižnic, s katerimi si bomo olajšali razvijanje svojega ogrodja. Ravno tako je za razvijanje aplikacij, namenjenih Cardboardu, Google razvil programsko knjižnico, ki aplikaciji omogoča enostaven prehod v stanje omogočanja navidezne resničnosti. Poleg tega sta obe napravi precej enostavni za uporabo, pri čemer se za delovanje Cardboarda potrebuje le pametni telefon, katerega pa ima dandanes že skoraj vsak Zemljan.

Poglavje 2

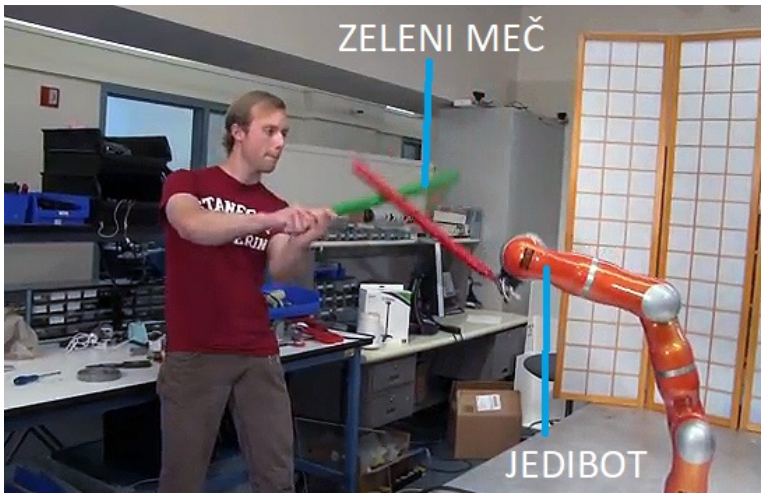
Pregled področja

Spekter aplikacij, namenjenih uporabi Kinecta, se je od njegove splavitve močno povečal, isto velja za število aplikacij, namenjenih navidezni resničnosti. Ti dve tehnologiji, ki sta bili na začetku namenjeni le neki osnovni ideji, sta sedaj segli tudi na veliko drugih področij. V nadaljevanju bomo opisali sorodne projekte ter predstavili uporabnost Kinecta in VR na različnih področjih.

2.1 Jedibot

Jedibot je projekt, ki so ga ustvarili študenti stanfordske univerze pri tečaju Eksperimentalne robotike [28, 50]. Robotska roka, Kinectov senzor za zaznavanje gibanja in dobro napisana programska oprema, je vse, kar so študentje potrebovali, da so ustvarili računalniško roko, ki vihti meč. Od tu tudi ime Jedibot. Robotska roka deluje tako, da ima v sebi zakodirane poteze za napad, pri obrambi pa ji pomaga Kinectov senzor za gibanje, ki sledi nasprotnikovemu zelenemu meču (senzor je nastavljen tako, da išče zelen meč) in posreduje informacije robotski roki, kako naj ta postavi svoj meč za najboljšo obrambo. Jedibot napada z naključno izbranimi potezami. Če ob udarcu pride do blokade, robot ponovno udari. V nasprotnem primeru preide v stanje obrambe. V tem stanju Jedibot uporablja Kinectov senzor, da prepozna zeleni meč iz ozadja, izvede analizo globine napadalčevega meča

glede na položaj svojega in se postavi na pravo mesto.



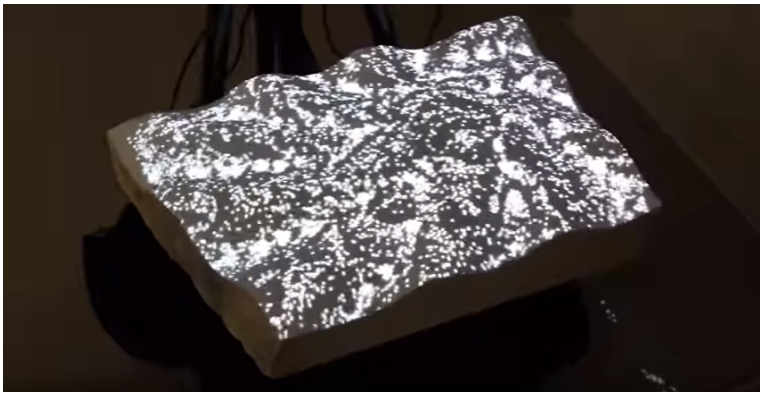
Slika 2.1: JediBot [19]

2.2 Kinect in HTC Vive

Projekt nizozemskih razvijalcev Jasperja Brekelmansa in Jeroena de Mooija združuje Kinect z VR platformo HTC Vive [25]. Prvi služi za sledenje teles dveh uporabnikov v 3D-prostoru, drugi pa za prikaz navideznega sveta. Informacije o položaju teles se nato uskladijo s pozicijskem sledenjem naprave HTC Vive. Tako je mogoče doseči, da uporabnika vidita videoprezentacijo sebe in drugega igralca v navidezni okolici.

2.3 Light fountain – a virtually enhanced stone sculpture

Light fountain [68] je projekt Franca Soline in Blaža Medena, ki združuje kamnito skulpturo in navidezno vodo v umetniško delo. Med delovanjem Kinect zaznava 3D-površino kamnite skulpture, na katero padajo navidezne



Slika 2.2: Kamnita skulptura, na katero so projicirane navidezne kapljice

kapljice. Te s pomočjo simulacije, ki uporablja podatke, dobljene iz Kinecta, o površini, odtekajo po skulpturi. Kapljice so na skulpturo projektirane s pomočjo video projektorja in so na podlagi prikazane kot svetlobne točke. Opazovalec lahko tako dobi občutek, kako se digitalna animacija po fizikalnih zakonih premika v realnem svetu.

2.4 Sledenje z več senzorji Kinect za usposabljanje vojaka

'Ko nizkocenovni senzorji videoiger postajajo vedno popularnejši, bi lahko interakcije, ki se v gospodinjstvih ustvarjajo med uporabniki in platformami, uporabili tudi za nizkocenovno aplikacijo usposabljanja vojakov,' je zapisano v članku [72] z naslovom *Multi-Kinect Tracking for Dismounted Soldier Training*. Ta govori o uporabi mnogo Kinectov, postavljenih na večje območje, kar omogoča vojakom svobodno gibanje ter možnost obračanja za 360 stopinj, medtem ko nosijo naglavni prikazovalnik.

Zakaj več Kinectov? Pri uporabi Kinecta je uporabnik omejen, saj mora ta biti obrnjen proti napravi, kar ustvari nenaraven uporabniški vmesnik. Problem je mogoče rešiti z dodajanjem dodatnih senzorjev v prostor, kar omogoča uporabniku svobodno gibanje. Zakaj naglavni prikazovalnik? V tra-

dicionalnih videoigrah je uporabnikov pogled fiksen, usmerjen proti televizijskemu ekranu. Za aplikacijo usposabljanja vojakov pa je možnost obračanje glave obvezna, zato vojaki nosijo naglavne prikazovalnike, kar jim omogoča obračanje glave znotraj področja usposabljanja.

2.5 Področje medicine in zdravstva

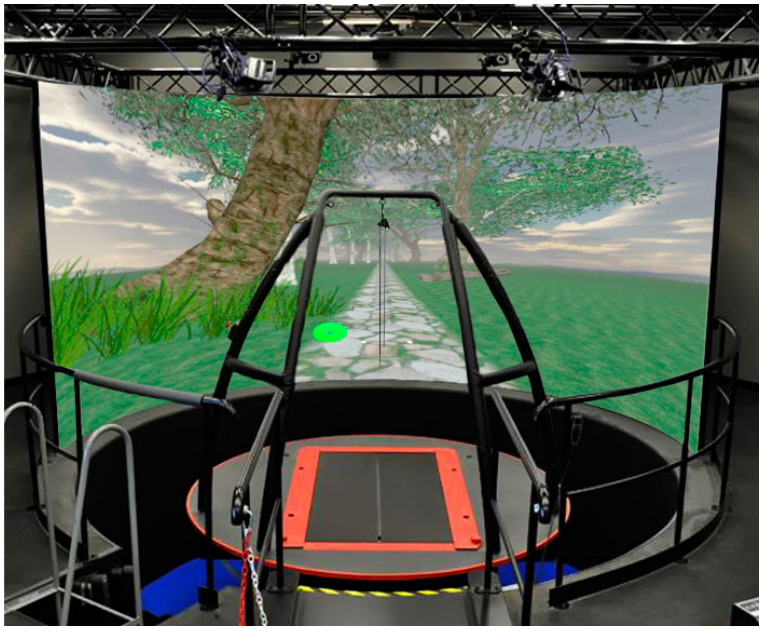
Zdravstvo in medicina morata stalno sprejemati nove in inovativne načine za napredek, da bi zdravstvenim delavcem zagotovila boljše usposabljanje in izobraževanje ter pacientom kvalitetnejše zdravljenje. Navidezna resničnost je v zdravstvu ena od pomembnejših inovacij, s pomočjo katere lahko zdravstveni delavci pridobijo nova znanja, izboljšujejo obstoječe veščine v varnem in kontroliranem okolju ter uporabijo VR za zdravljenje bolnikov.

2.5.1 Usposabljanje zdravnikov in medicinskih sester

Navidezna resničnost ima velik pomen pri izobraževanju in usposabljanju zdravstvenega osebja. Usposabljanje zdravnikov in medicinskih sester za izvajanje rutinskih postopkov je dolgotrajno, običajno pa ga je treba opraviti z drugimi zaposlenimi in dragimi strokovnjaki. Zaradi tega se navidezna resničnost vedno bolj uporablja za učenje anatomije, prakse in kontrole okužb. Biti postavljen v realistično simulacijo postopka in prakso korakov in tehnik je veliko boljše usposabljanje kot gledanje videoposnetka ali stanje v sobi poleg strokovnjaka in opazovanje njegovega dela. Velik pomen ima tudi v operacijskih sobah, saj omogoča kirurgu, da se ta s pomočjo VR na operacijo dobro pripravi. Ponuja pa tudi možnost ostalim, da operacijo opazujejo od blizu. VR ponuja zdravnikom tudi možnost, da doživijo simptome svojih pacientov. Startup Embodied Labs je tako ustvaril aplikacijo We Are Alfred, ki ti ustvari svet, v katerim izkusiš, kako je biti star in imeti težave z vidom in sluhom [6, 3].

2.5.2 Rehabilitacija možganske kapi

Navidezna resničnost postaja nov pristop pri rehabilitaciji možganske kapi. Prednost takega zdravljenja je predvsem možnost izvajanja aktivnosti, ki jih je drugače nemogoče izvajati izven kliničnega območja. Poleg tega so VR programi bolj zanimivi in prijetni kot tradicionalne terapije in s tem spodbujajo večjo število ponovitev vaj. Eden izmed takih sistemov je CAREN (Computer Assisted Rehabilitation Environment), ki v svoji VR-dejavnosti bolniku pomaga na psihološki, fizični in kognitivni ravni [70, 63].



Slika 2.3: CAREN [62]

2.5.3 Aplikacija za pomoč bolnikom z multiplo sklerozo

VR v medicini uporabljajo pri zdravljenje bolnikov z multiplo sklerozo (angl. Multiple sclerosis - MS). En izmed načinov je aplikacija 'Shark Punch' [65, 61], katere avtor je izredni profesor s teksaške univerze John Quarles, ki je bil

tudi sam diagnosticiran z MS. Veliko ljudi z MS se pri vadbi pregreje, zato je zanje primerna vodna terapija, ki jim pomaga obdržati nizko temperaturo in ravnotežje. Tukaj pride na vrsto tudi aplikacija Shark Punch, ki uporabniku vadbo spremeni v zabavo. Ta si namesti VR-očala, dihalko in se potopi. Okoli njega začne plavati navidezni morski pes, ki poskuša uporabnika ugrizniti. Njegova naloga je, da udari morskega psa, ko se ta zadosti približa, in ga tako odžene stran.

2.5.4 Upravljanje bolečine pri žrtvah opeklin

Za žrtve opeklin je bolečina stalen problem, zdravniki pa verjamejo, da bi lahko odvračalna terapija s pomočjo navidezne resničnosti žrtvam pomagala prenašati bolečino. Univerza v Washingtonu je za ta problem ustvarila VR-aplikacijo imenovano SnowWorld, v kateri igralec meče snežene kepe v pingvine in posluša pomirjujočo glasbo.

Slika 2.4: Aplikacija SnowWorld [48]

VR-aplikacija pri pacientu ustvari iluzijo, da je ta v drugem svetu, s tem pa mu zmanjša percepcijo za zaznavanje bolečine. Tako bi bila opravila, kot sta fizična terapija in oskrba ran, manj boleča. Študija iz leta 2011 je pokazala, da je vojakom s poškodbami opeklin uporaba aplikacije SnowWorld bolj pomagala kot morfij [64, 1].

2.5.5 Zdravljenje PTSD

V zadnjem času vedno več klinik uporablja navidezno resničnost za pomoč vojnim veteranom pri zdravljenju posttravmatske stresne motnje (angl. Posttraumatic stress disorder - PTSD). S pomočjo navidezne resničnosti in simulacij vojn veterane v kontroliranem in varnem okolju nadzorujejo in jih učijo premagovati strahove in slabe občutke, ki jih imajo ob podoživljanju travmatičnih dogodkov, ki so se jim zgodili v preteklosti [1].

2.5.6 Vpliv Microsoftovega Kinecta na rehabilitacijo in fizično terapijo

V letu 2014 sta Hossein Mousavi Hondori in Maryam Khademi objavila članek [69], v katerem je predstavljen tehnični in klinični učinek Microsoftovega Kinecta v zdravstvu. Ta zajema študije o bolnikih z nevrološkimi motnjami, vključno s kapjo, Parkinsonovo [4], cerebralno paralizo in MS.

Ena izmed študij je uporaba Kinecta za rehabilitacijo hoje. Za ta namen je bila ustvarjena aplikacija, v kateri uporabnik stoji pred ekranom, njegovo premikanje pa spremlja senzor Kinect. Na ekranu je prikazan par navideznih čevljev, ki ponazarjajo bolnikova stopala. V igri mora uporabnik premikati stopala na navidezne objekte, ki so prikazani na ekranu, ter se umikati navideznim oviram.

Članek opisuje tudi vpliv uporabe Kinecta pri zdravljenju starejših bolnikov. Študije so pokazale, da je Kinect uporaben za izboljšavo statičnih in dinamičnih funkcij ravnotežja in s tem pripomore starejšim pri preprečevanju padca in njihovi gibljivosti. Tako rezultati študij razkrivajo vedno večje zanimanje za uporabjo Kinecta v medicinske namene.

Poglavje 3

Orodja in okolja

3.1 Navidezna resničnost

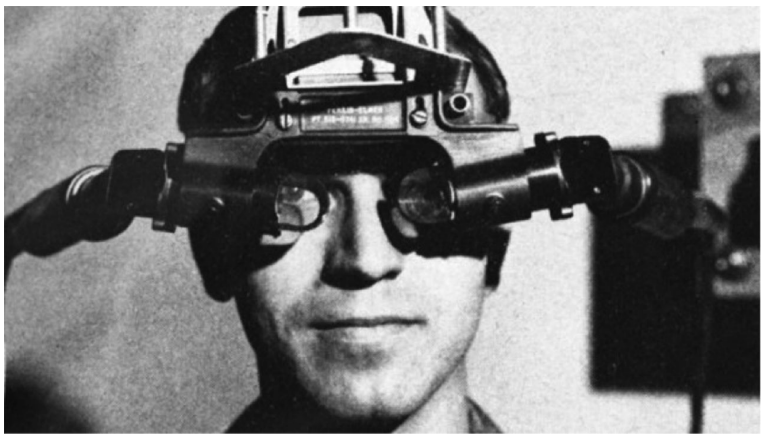
Navidezna resničnost je povzročila nastanek mnogo platform in razvojnih okolij. Prve omembe navidezne resničnosti segajo že v sredino devetnajstega stoletja, a v malce drugačnem smislu, kot jo uporabljamo danes. Francoski dramatik Antonin Artaud je želel s svojimi avantgardnimi deli izbrisati razlike med iluzijo in realnostjo, hotel je, da obiskovalci njegovih predstav pozabijo na realnost in se vživijo v predstavo. Želel je, da jim njegova predstava postane nekakšna nova resničnost - navidezna resničnost.

Ideja za navidezno resničnost, kot pa jo poznamo danes, se je rodila v tridesetih letih prejšnjega stoletja, z objavo zgodbe *Pygmalion's Spectacles*, ki jo je napisal ameriški pisatelj znanstvene fantastike Stanley G. Weinbaum. Zgodba vsebuje idejo, ki govori o očalih, skozi katera lahko doživiš fikcijski svet. Mnogi so se nato začeli ukvarjati z napravami, ki so spodbujale človekove čute, začeli so izumljati zaslone, ki se jih je bilo mogoče namestiti na glavo, ter druge pripomočke, ki bi pripomogli k doživljaju navideznega sveta [60]. Pomembnejši trenutki v razvoju navidezne resničnosti [13]:

- 1950 – Morton Heiligova Sensorama: Arkadno-teatrični kabinet, ki je uporabniku spodbudil vse čute.

Slika 3.1: Naprava Sensorama [46]

- 1960 – Nastanek prvega naglavnega prikazovalnika.
- 1961 – Headsight: Prva HMD naprava z zaznavanjem gibanja.
- 1968 – Sword of Damocles: Prvi VR naglavni prikazovalik, ki je deloval s pomočjo računalnika, in ne več s pomočjo kamere kot prejšne naprave.



Slika 3.2: Sword of Damocles [51]

- 1987 – Začela se je uporabljati besedna zveza 'navidezna resničnost'.

- 1991 – Virtuality Group arkadne naprave: Podjetje The Virtuality Group je izdalo več arkadnih naprav in iger, ki so podpirale VR. VR je tako postal dostopen širši publiki.

- 1993 – SEGA je objavil razvoj novih VR-očal.

- 1995 – Nintendo Virtual Boy: Prva prenosna konzola, ki je lahko prikazala pravo 3D-grafiko.

Slika 3.3: Nintendov Virtual Boy [59]

V enaindvajsetem stoletju pa sta se napredek in razvoj navidezne resničnosti še povečala. Google je ustvaril Street View, Palmer Luckey je osnoval prvi prototip Oculus Rifta. Vzpon pametnih telefonov je navidezni resničnosti ponudil nove trge in nove možnosti za razvoj. Veča se ponudba platform navidezne resničnosti, cena za dosego VR pa se hkrati manjša in je tako veliko bolj dostopna kot pa v prejšnjih letih. S pomočjo navideznega sveta se danes zabavamo, se učimo, poslujemo, navidezna resničnost praktično prodira na vsa področja.

3.2 Kinect

Kinect [73] je skupek senzorjev za zaznavanje gibanja in govora, ki ga je razvilo podjetje Microsoft. Ime 'Kinect' izhaja iz besed kinetic in connect [21]. S tem je Kinectov ustvarjalec želel izraziti glavne lastnosti svojega izdelka: gibanje in povezljivost. Kinect je ob svojem prihodu leta 2010 služil dopolnitvi igralniške izkušnje na platformi Xbox 360, pozneje pa se je njegova namembnost močno razširila še na druga področja. Do zdaj poznamo tri verzije senzorja Kinect:

- Kinect for Xbox 360,
- Kinect for Xbox One,
- Kinect for Microsoft Windows.

Slika 3.4: Kinect for XBox 360 [30]

Pri razvijanju diplomske naloge smo za prvo aplikacijo na začetku uporabljali Kinect for Xbox 360, pozneje pa smo zaradi boljše natančnosti in zmogljivosti prešli na novejšo napravo Kinect for Xbox One. Za drugo aplikacijo diplomske pa smo uporabili oba Kinecta, saj je aplikacija namenjena dvema igralcema, ki se lahko povežeta preko skupnega omrežja, in sta zaradi tega potrebni dve napravi za zajemanje gibanja.

Slika 3.5: Kinect for XBox One [24]

3.2.1 Komponente in primerjava senzorjev Kinect for XBox 360 in Kinect for XBox One

Globinski senzor: Globinski senzor Kinecta for XBox 360 je proizvod izraelskega podjetja PrimeSense in je sestavljen iz IR-projektorja ter IR-senzorja (CMOS). Deluje na tehnologiji svetlobnega kodiranja (light coding) oziroma na principu projekcije vzorcev, kjer infrardeči projektor projicira znan pikčasti vzorec na objekte okoli sebe. IR-senzor ta vzorec zaznava in zajema intenziteto IR-pik, ki padajo na okolico. Zaradi tega je uporaba Kinecta boljša v zaprtih prostorih, saj bi drugače sončna svetloba zamazala vzorec pik. Slika, ki jo senzor zajame, je tako sestavljena iz istih infrardečih podatkov, kot jih je projiciral IR-projektor. Če primerjamo to sliko s projiciranim vzorcem iz projektorja, pride zaradi razdalje med senzorji ter oddaljenosti predmeta od Kinecta do odmika ali neskladja med vzorcema. [55] Kinect lahko tako s triangulacijo izmeri globino slike. Globinska slika ima ločljivost 320 x 240 slikovnih pik, vidno polje (FoV) tega senzorja meri 58,5 x 46,6 stopinje, kar se izraža v povprečju 5 x 5 slikovnih pik na stopinjo [26, 71].

Kinect for XBox One je v celoti proizveden s strani Microsofta, njegov globinski senzor pa je sestavljen iz več IR-oddajnikov in IR-kamere. Ta meri

globino na principu 'time of flight' (ToF), kjer se globina določi glede na čas, ki ga svetloba potrebuje od oddajnikov do objekta in nazaj. Zato senzor neprestano oddaja IR-svetlobo z moduliranimi valovi in zaznava časovne zamike prihajajočih žarkov. Senzor ima globinsko sliko ločjivosti 512 x 424 slikovnih pik z vidnim poljem 70,6 x 60 stopinj, kar doprinese k povprečju 7 x 7 slikovnih pik na stopinjo [26, 71].

Kamera RGB: Kamera zajema barvno sliko, sestavljeno iz zaznanih rdečih, zelenih in modrih komponent okolice, od tukaj tudi ime kamera RGB. Kinect for XBox 360 zajema slike s hitrostjo 30 Hz v ločljivosti 640 x 480 slikovnih točk in vidnim poljem 62 x 48,6 stopinje, omogoča pa tudi zajemanje slik v ločljivosti 1280 x 1024, ampak se pri tem hitrost zajemanja slik zmanjša. Kamera Kinecta for XBox One je boljša, saj pri hitrosti 30 Hz zajema slike v ločljivosti 1920 x 1080 ter ima vidno polje, ki meri 84,1 x 53,8 stopinje[26] [71]. Kamera pripomore pri prepoznavanju obrazov in drugih zaznavanjih.

Polje mikrofonov: To je vrsta mikrofonov, ki so postavljeni na spodnjo stran senzorja Kinect. Služijo glasovnim ukazom, prepoznavanju govora, lociranju njegovega izvora ter omogočajo uporabniku, da izolira svoj glas pred ostalimi hrupi v sobi [15].

Ostale razlike: Kinect for XBox 360 ima v nasprotju s Kincetom for XBox One stojalo z motorčkom za spremembo naklona. Novejši Kinect pa ima pred starejšo verzijo druge prednosti. Pri zaznavanju teles lahko v nasprotju s prejšnjo verzijo, ki lahko hkrati sledi le dvema osebama, ta sledi kar šestim. Boljše zna definirati okostje človeka, saj definira kar 26 človeških sklepov, v nasprotju s starejšo različico, ki definira le 20 [14].

3.3 Cardboard VR

Cardboard VR je platforma navidezne resničnosti, ki jo je ustvaril Google leta 2014 in ki je dobila ime po svoji prepogljivi papirnati obliki [12]. Za uporabo

platforme je potrebno samo vstaviti svojo mobilno napravo na zadnjo stran Cardboard VR-a in gledati vsebino skozi leči. 3.6

Slika 3.6: Google Cardboard [47]

3.4 Naprave za navidezno resničnost

Seveda pa ni Cardboard VR edina platforma za predvajanje oziroma doživljanje navidezne resničnosti.

HTC Vive: Je naglavni prikazovalnik navidezne resničnosti, ki ga je razvil HTC v sodelovanju s podjetjem Valve. Platforma uporablja 'room scale' tehnologijo sledenja, kar omogoča uporabnikom, da se premikajo po 3D-prostoru in uporabljajo ročne krmilnike za interakcijo z okolico. Naprava uporablja dva zaslona z ločljivostjo 1080 x 1200 in hitrostjo osveževanja 90 Hz. Vsebuje dve brezžični IR 'Lighthouse' kameri, ki sta postavljeni v kotih sobe, ter 70 IR-senzorjev: 32 na naglavnem prikazovalniku ter 19 na vsakem ročnem krmilniku. Rezultat tega je, da se lahko uporabnik prosto giblje po prostoru in v njem zaznava gibljive ali statične predmete. [17] [58] Naprava ima mnogo prednosti: Zanjo je ustvarjenih že mnogo inovativnih iger, ponuja intuitiven vmesnik ter eno izmed najboljših izkušenj navidezne resničnosti. Sledenje

glavi je boljše kot pri konkurenčni napravi Oculus Rift, omogoča igralcem, da hodijo v navideznem svetu ter ponuja ročne krmilnike 6DOf, ki so prilagodljivi in enostavni za uporabo. Slabosti platforme so, da nima vgrajenega zvoka, ostati mora povezana z računalnikom, da ima dolge žice ter veliko opreme, za kar se potrebuje precej časa in prostora za pripravo platforme, in visoka cena [16].

Slika 3.7: HTC Vive [18]

Oculus Rift: Je platforma navidezne resničnosti, ki jo je razvil Oculus VR, oddelek družbe Facebook Inc. Vsebuje zaslon OLED ločjivosti 2160 x 1200, s hitrostjo osveževanja 90 Hz in z vidnim poljem 110 stopinj. Ima integrirane slušalke, ki zagotavljajo zvočni učinek 3D, ter rotacijsko in pozicijsko sledenje. Sistem za pozicijsko sledenje, imenovan 'Constellation', je zagotovljen s pomočjo infrardečega senzorja USB, ki sprejema svetlobo, ki prihaja iz oddajnikov, vgrajenih v naglavnem prikazovalniku. Platforma se lahko nadgradi s krmilnikom gibanja, imenovanim Oculus Touch. Ta je sestavljen iz dveh enot, ki ju uporabnik drži v rokah. Krmilnika sta v sistemu zaznana s pomočjo sistema 'Constellation' in sta tako lahko prikazana v navidezni

resničnosti [5] [35]. Prednost naprave je vgrajeni zvočni sistem 3D, ki naredi izkušnjo navidezne resničnosti še toliko bolj resnično, udoben in lahek naglavni prikazovalnik, funkcionalni sistem s širokim izborom razpoložljivih iger ter obsežno in natančno območje sledenja. Slabosti Oculus Rifta so, da potrebuje zmogljiv računalnik za uporabo, možnost izhajanja zvoka iz slušalk, ročne krmilnike je treba dodatno kupiti ter nekoliko visoka cena [36].

Slika 3.8: Oculus Rift [37]

Samsung Gear VR: Podjetje Samsung je v letu 2015 izdalo nov naglavni prikazovalnik, imenovan Samsung Gear VR, ki je zasnovan za delo s Samsungovimi vodilnimi telefoni [45]. Ko je v platformi uporabljena kompatibilna Samsungova naprava, telefon deluje kot prikazovalnik in procesor, Gear VR pa deluje kot krmilnik, ki ima svoje vidno polje, IMU (inertial measurement unit) za rotacijsko sledenje, gumbe za upravljanje aplikacij ter senzor bližine, da zazna, kdaj je naglavnik v uporabi [10]. Prednost naprave je ta, da je popolnoma brezžična, udobna in enostavna za uporabo ter dobavljiva po sorazmerno nizki ceni. Slabosti so omejenost prilagajanja leč, povzroča lahko slabost, še vedno ima omejen obseg aplikacij, glavna slabost, zaradi

katere se nismo odločili za to napravo, pa je ta, da je naprava združljiva le s peščico Samsungovih telefonov [44].

Slika 3.9: Samsung Gear VR [11]

Sony PlayStation VR: Project Mopheus, zdaj PlayStation VR, je naglavni prikazovalnik navidezne resničnosti, ki ga je razvilo podjetje Sony Interactive za delovanje z igralno konzolo PlayStation 4 [49]. Naprava vsebuje zaslon OLED z ločljivostjo 1920 x 1080, vidnim poljem 100 stopinj in hitrostjo osveževanje 120 Hz. Ima procesorsko omarico, ki omogoča video izhod za televizijo in obdelavo 3D zvočnega efekta ter možnost 6DoF (Six degrees of freedom) sledenja glavi [41, 39]. Prednosti te platforme so: zanimiv dizajn, udobna za uporabo, dober izbor aplikacij, ponuja odlično VR-izkušnjo ter skoraj dosega kvaliteto delovanja PC-ja. Slabosti PlayStation VR-a so te, da pri nakupu očal ne prejmeš zraven še ostalih pripomočkov, ki jih potrebuješ za doživetje VR-a, naprava ni tako močna, kot sta napravi Oculus Rift in HTC Vive, ter je mogoč pojav težav pri zaznavanju in obračanju igralca [40].

Cardboard VR je sestavljen iz recikliranega papirja, njegove leče pa so narejene iz plastike. Zaradi nizkocenovnih materialov in precej enostavne izdelave produkta je cena izdelka nizka. Cardboard VR pa je tudi ena izmed najpreprostejših platform navidezne resničnosti za nastavitev in uporabo. Svoj telefon postaviš v napravo in že si pripravljen na VR-doživetje [42]. Prej

Slika 3.10: Sony PlayStation VR [38]

smo lahko videli, da je mnogo platform navidezne resničnosti namenjeno le določenim napravam, pri Cardboardu pa je prednost ta, da lahko deluje skoraj z vsemi pametnimi telefoni, ki podpirajo VR. Zaradi teh razlogov smo se odločili, da bo platforma navidezne resničnosti, s katero bomo razvijali našo aplikacijo, Google Cardboard VR.

3.5 Ostala orodja in okolja

3.5.1 Kinect for Windows SDK

Kinect for Windows SDK je zbirka gonilnikov, knjižnic in preprostih primerov, s pomočjo katerih lahko uporabniki razvijejo aplikacije, primerne za Kinect. Zbirka je podprta na operacijskih sistemih Windows. Leta 2011 jo je izdal Microsoft, od takrat pa se je zbirka posodabljala in tako je nastalo več verzij [21].

Za svojo aplikacijo smo najprej uporabili paket Kinect for Windows 1.8, saj je ta omogočal uporabo naprave Kinect for Xbox 360. Pozneje, ko smo uporabili Kinect for Xbox One, nam ta paket ni več omogočal delovanja novejšega Kinecta, zato smo morali namestiti še zbirko Kinect for Windows 2.0, ki podpira razvoj aplikacij na novejši napravi.

3.5.2 Visual Studio

Visual Studio je razvojno okolje, ki ga je ustvarilo podjetje Microsoft. Orodje omogoča razvoj programov za operacijske sisteme Windows, spletnih storitev, spletnih aplikacij ... Visual Studio sam po sebi ne podpira nobenega programskega jezika, rešitve ali orodja, omogoča pa dodajanje funkcionalnosti, kodirane kot VSPackage. Ko je paket nameščen, je funkcionalnost na voljo kot storitev. Visual Studio vsebuje naslednje komponente:

- Code editor,
- Debugger,
- Designer,
- ostala orodja.

Leta 1997 je izšla prva verzija, ki se je imenovala Visual Studio 97. Tej je sledilo še 10 verzij [29]. Najnovejša je trenutno Visual Studio 2017, svojo aplikacijo pa smo razvijali v Visual Studio 2015 ter v Visual Studiu 2017.

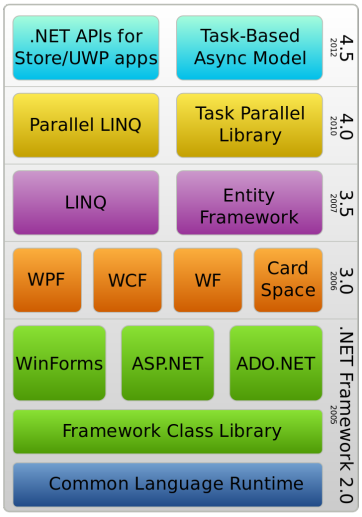
3.5.3 .NET Framework

Microsoft .NET je ogrodje za razvijanje programske opreme, ki teče pretežno na operacijskih sistemih Microsoft Windows [33]. Ogrodje .NET je skupek dveh komponent.

'Framework Class Library (FCL)' [34] je obsežna knjižnica razredov, vmesnikov in vrednostnih tipov, ki omogoča dostop do sistemskih funkcij, zagotavlja uporabniški vmesnik, dostop do podatkov, povezljivost baze podatkov, kriptografijo, razvoj spletnih aplikacij, numerične algoritme in omrežne komunikacije. Je temelj, na katerem se gradijo .NET aplikacije, ter skrbi za medopravilnost jezikov. Programerji izdelujejo programsko opremo tako, da kombinirajo svojo izvorno kodo z .NET Framework in drugimi knjižnicami.

Druga komponenta je 'Common Language Runtime (CLR)' [8], ki skrbi za izvajanje programov v okolju .NET, upravlja kodo in zagotavlja storitve,

ki olajšajo razvojni proces. Pripomočki in orodja prikazujejo funkcionalnost CLR-ja in omogočajo pisanje kode, ki ima v tem izvajalnem okolju mnogo koristi. Koda, ki se razvija z jezikovnim prevajalnikom, se imenuje upravljana koda ali 'managed code'. Ta ima koristi od funkcionalnosti, kot so integracija vmesnih jezikov, upravljanje z izjemami vmesnih jezikov, izboljšana varnost, podpora za postavitev projektov, poenostavljen model za interakcijo med komponentami ter storitve pregledovanja in odpravljanja napak.



Slika 3.11: .NET Framework [32]

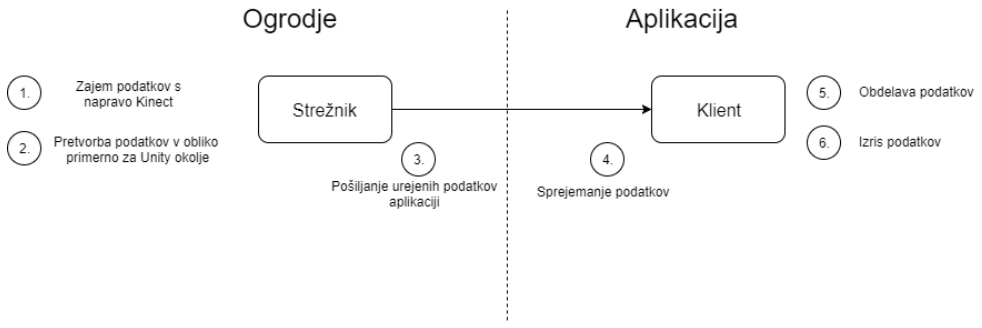
3.5.4 Unity

Unity je igralni pogon, ki ga je razvil Unity Technologies leta 2005. Je sistem, ki nam olajša gradnjo računalniških iger in drugih interaktivnih vsebin. Sistem podpirata Windows OS ter Mac OS X, na katerih lahko gradimo aplikacije, ki bodo delovale na platformah, kot so Android, Apple TV, BlackBerry 10, iOS, Linux, Nintendo 3DS line, macOS, PlayStation 4, PlayStation Vita, Unity Web Player, Wii, Wii U, Nintendo Switch, Windows Phone 8, Windows, Xbox 360 in Xbox One [57].

Poglavje 4

Razvoj aplikacije

Kot smo povedali že v uvodu, je cilj naše diplomske vzpostaviti delujoč sistem, ki združuje sposobnost zaznavanja človekovega gibanja in prikaz teh podatkov v navidezni resničnosti. Ustvariti ogrodje, ki bo podatke skeletov, zaznanih s pomočjo senzorja Kinect, pretvorilo v obliko, ki je primerna za delovanje v okolju Unity, in jih nato pošiljalo želenim aplikacijam. Zadnji korak je razvoj dveh takih aplikacij, ki jih je s pomočjo Google Cardboarda mogoče doživeti skozi izkušnjo navidezne resničnosti: Hoja nad prepadom in Rokoborba. Prva je namenjena enemu igralcu, ki s pomočjo Kinecta v navidezni resničnosti poskuša ostati na palici in ne pasti v prepad. V drugi aplikaciji, ki je namenjena dvema igralcema, pa se soigralca pomerita v igri rokoborbe.



Slika 4.1: Osnovna shema našega sistema

4.1 Postavitev delovnega okolja

Razvoj aplikacije je potekal na osebnem računalniku, s specifikacijami:

- operacijski sistem: Windows 10 Home x64,
- procesor: Intel core i7-4790 3.60 GHz,
- grafična kartica: NVIDIA GeForce GTX 760/Intel HD Graphics 4600,
- kapaciteta pomnilnika RAM: 16 GB DDR3,

ter na prenosnem računalniku z naslednjimi specifikacijami:

- operacijski sistem: Windows 10 Pro x64,
- procesor: Intel core i7-6700HQ 2.60 GHz,
- grafična kartica: NVIDIA GeForce GTX 960M/Intel HD Graphics 530,
- kapaciteta pomnilnika RAM: 16 GB DDR4.

Poleg računalnikov, na katerih smo razvijali aplikacijo, smo za dosego cilja potrebovali še dve napravi Kinect ter naglavni prikazovalnik. Te smo si izposodili od Laboratorija za računalniško grafiko in multimedije. Kinecta sta bila različnih verzij, saj je bil eden namenjen konzoli XBox 360, drugi pa konzoli XBox One. Zaradi tega sta potrebovala tudi drugačno programsko opremo. Na računalnik, na katerega je bil povezan starejši Kinect for XBox 360, smo namestili programsko opremo Kinect for Windows SDK v1.8, ki jo je mogoče prenesti z Microsoftove uradne strani [23]. Računalnik, povezan z novejšim Kinectom, je za pravilno delovanje senzorja zaznavanja potreboval Kinect for Windows SDK v2.0 programsko opremo [22]. Namestitveni postopek je pri obeh precej podoben in enostaven. Med namestitvijo programske opreme mora biti senzor Kinect ves čas izklopljen iz računalnika.

Za razvoj aplikacije smo uporabili Microsoft Visual Studio, ki ponuja brezplačni razvoj aplikacij v jeziku C#, ter Unity za kreiranje videoigre. Jezik C# nam je ustrezal, saj je celoten Kinectov SDK napisan v njem,

primeren pa je tudi za razvijanje aplikacij v Unityju. Microsoftovo integrirano razvojno okolje (IDE) Visual Studio je mogoče prenesti z njihove uradne strani [66]. Razvojno okolje Unity pa je mogoče naložiti z naslednje strani [56]. Na strani je mogoče izbirati med tremi različnimi namestitvenimi paketi, mi smo izbrali verzijo Personal, saj je ta brezplačna za uporabo.

4.2 Razvoj ogrodja

Za razvoj ogrodja smo morali najprej implementirati zaznavanje gibanja z napravo Kinect. Ko je bil ta del opravljen, smo začeli s pretvorbo teh podatkov v obliko, ki je primerna za delovanje v okolju Unity. Na koncu smo morali implementirati še proces, ki je bil zadolžen za pošiljanje urejenih podatkov končni aplikaciji. Aplikacija je prejete podatke obdelala in jih izrisala. Res je, da se ta proces zgodi na mobilni napravi znotraj aplikacije, ampak ker je implementacija izrisovanja človekovega gibanja enaka tako v aplikaciji Hoja nad prepadom kot v aplikaciji Rokoborba, jo bomo predstavili že v tem oddelku.

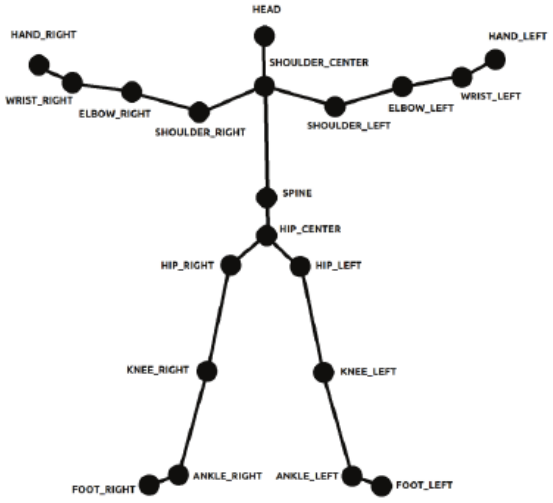
4.2.1 Zaznavanje gibanja s senzorjem Kinect

Implementirati zaznavanje gibanja v naše ogrodje ni bilo težavno, saj smo si pri tem pomagali z Microsoftovo knjižnico Kinect for Windows. Ta je vsebovala projekt Body Basics, ki je namenjen prikazovanju človeškega skeleta in ima implementirano zaznavanje gibanja s Kinectom. S preučevanjem projekta smo ugotovili, katere funkcije bi potrebovali v svojem ogrodju in katere so za nas neuporabne. V Visual Studiu smo tako ustvarili nov projekt in vanj prenesli uporabne funkcije, ki so bile potrebne za zaznavanje človekovega gibanja. Zaznavanje je v tem trenutku delovalo tako, da se je ob zagonu projekta zagnal tudi senzor Kinect. Ta je v vsakem trenutku beležil vizualne okvirje in jih posredoval funkcijam, ki so bile zadolžene za branje teh okvirjev. Če znotraj okvirja ni bilo zaznanega nobenega skeleta, se je ta okvir zavrgel. V primeru zaznanega telesa pa smo začeli odčitavati informacije o

njegovih sklepih. Ker je projekt Body Basics namenjen le prikazovanju skeleta, in ne shranjevanju podatkov, smo morali kreirati objekt, v katerem smo hranili informacije o skeletu. Objekt je tako vseboval informacijo o položaju skeleta ter podatke o sklepih telesa.

4.2.2 Pretvorba Kinectovih podatkov za uporabo v okolju Unity

Objekt, v katerem smo hranili informacije o skeletu, je vseboval položaj telesa ter pozicije sklepov. V okolju Unity je za pravilno izrisovanje modela potrebna rotacija, ki za vsak sklep telesa pove, kako je ta postavljen oziroma obrnjen glede na svojega predhodnika (s slike 4.2 je razvidno, da je ELBOW_LEFT predhodnik sklepa WRIST_LEFT, ta pa je predhodnik sklepa HAND_LEFT).



Slika 4.2: Skelet telesa, kot ga zazna Kinect [53]

Zaradi tega smo morali pozicijske podatke sklepov pretvoriti v rotacijske. To smo storili s pomočjo paketa Kinect with MS-SDK, ki smo ga naložili z Unityjeve uradne strani [27]. Paket Kinect with MS-SDK je sestavljen iz niza

primerov uporabe Kinecta ter skript, ki Kinectu omogočajo različne funkcionalnosti znotraj okolja Unity. Iz skript smo tako v svoj projekt prenesli tiste funkcije, ki so sodelovale pri pretvorbi pozicije sklepov v rotacije, in tiste, ki so Kinectove podatke optimizirale za uporabo znotraj Unityjevega okolja. Prenesene funkcije je bilo nato treba popraviti oziroma prilagoditi našemu razvojnemu okolju, saj so te delovale s pomočjo knjižnic, ki so podprte le znotraj okolja Unity. Med pretvorbo smo iz pozicijskih podatkov najprej ustvarili rotacijsko matriko, ki smo jo pozneje pretvorili v kvaternion (objekt znotraj Unityjevega okolja, namenjen prikazu rotacij). Za izračun orientacije izbranega sklepa smo potrebovali podatke o položaju njegovega 'starša' in 'potomca'. Na podlagi položaja izbranega sklepa ter položaja starša smo definirali in izračunali 'nevtralno' orientacijo sklepa (npr. iztegnjena roka). Nato smo s pomočjo položaja izbranega sklepa in položaja potomca izračunali razliko v orientaciji sklepa glede na nevtralni položaj (t.i. lokalna orientacija). Te rotacijske podatke smo v naslednjih korakih prenesli na orientacije ustreznih GameObject skeletnih objektov možica, ki morajo imeti ustrezno hierarhijo in nevtralne orientacije.

4.2.3 Vzpostavitev povezave UDP med strežnikom in odjemalcem

Pretvorjene podatke smo nato morali s strežnika pošiljati odjemalcu. Preizkusili smo dve možnosti: TCP ali Transmission Control Protocol in UDP ali User Datagram Protocol [52].

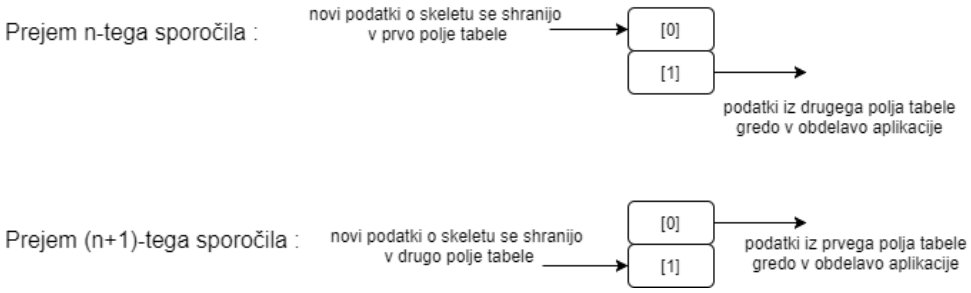
TCP je protokol, ki temelji na vzpostavljeni povezavi med dvema napravama, zato se sporočila med njima prenašajo zanesljivo v obe smeri, so brez napak, podvojevanja in v pravem vrstnem redu. TCP je primeren za aplikacije, ki zahtevajo visoko zanesljivost, čas prenosa pa je relativno manj kritičen. Uporabljajo ga protokoli, kot so: HTTP, HTTPS, FTP, SMTP, Telnet. Spada pod 'težke' protokole, saj za nastavitev vtičnice oz. vzpostavitev povezave zahteva tri pakete. TCP izvaja preverjanje napak in obnovitev napak. Napačni paketi se ponovno prenesejo od vira do naslovnika.

Protokol UDP je veliko bolj preprost in zaradi tega tudi hitrejši od prej omenjenega protokola TCP. Primeren je za aplikacije, ki potrebujejo hiter in učinkovit prenos, npr. videoigre. Protokoli, ki ga uporabljajo, so: DNS, DHCP, TFTP, SNMP, RIP in VOIP. Protokol ne skrbi za urejanje paketov, saj so pri tem protokolu paketi neodvisni drug od drugega. Veliko pripomore k hitrosti tudi to, da protokol ne ponuja obnavljanja napačnih paketov ter ne zagotavlja, da bodo sporočila prišla do naslovnika. Deluje po pravilu najboljšega napora.

V našem projektu smo preizkusili obe možnosti pošiljanja podatkov in se na koncu odločili za protokol UDP, saj smo želeli, da prejete podatke, ki smo jih dobili iz Kinecta, kar se da najhitreje pošljemo do odjemalca. Tako smo na strežniku s pomočjo Microsoftovih knjižnic in razredov UdpClient in IPEndPoint ustvarili točko, ki je povezani napravi pošiljala podatke. Da pa smo lahko pošiljali podatke prek omrežja, smo morali te pred pošiljanjem serializirati in jih iz objektov pretvoriti v tabelo bajtov. To nam je omogočila Newtonsoftova knjižnica Json.NET [20].

4.2.4 Prejemanje podatkov in njihovo shranjevanje

Implementirali smo skripto, ki je bila odgovorna za prejemanje in shranjevanje Kinectovih podatkov, ki jih je dobila od strežnika. Na odjemalčevi strani se je s pomočjo knjižnic ustvarila točka UDP, ki se je povezala s strežnikom in asinhrono sprejemala njegove podatke. IP-naslov, na katerem je gostoval strežnik je uporabnik ročno vpisal ob zagonu aplikacije. Odjemalec je prejete podatke deserializiral in jih ponovno pretvoril v objekte, s katerimi je lahko delal naprej. Objekte je izmenično zapisoval v tabelo 4.3, ki je vsebovala dve polji. S tem smo želeli preprečiti, da bi se podatki, ki še niso bili obdelani s strani aplikacije, uničili.

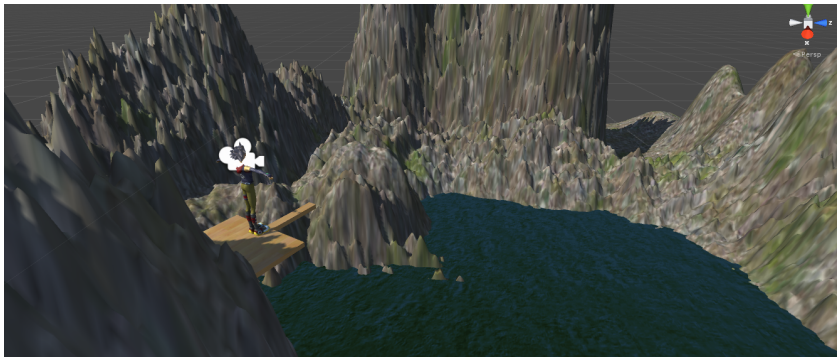


Slika 4.3: Odjemalčevo shranjevanje podatkov v tabelo

4.2.5 Gibanje v Unityju

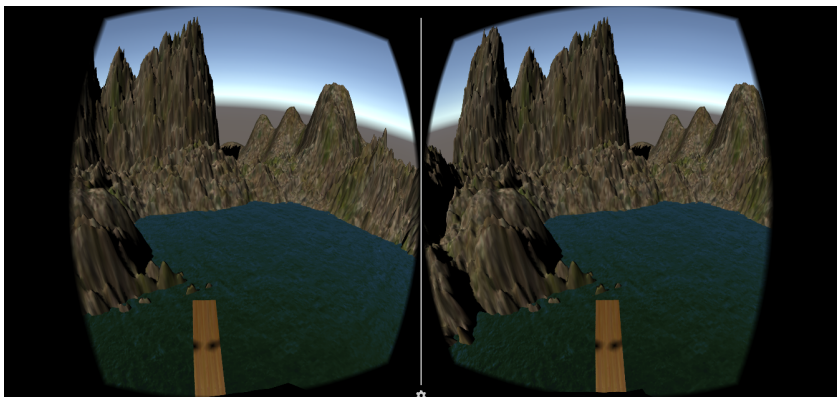
Paket Kinect with MS-SDK [27], ki je bil že prej omenjen, je vseboval tudi skripto Kinect AvatarController. Ta je mapirala Kinectove sklepe na posamezne dele telesa izbranega možica in skrbela za njegovo izrisovanje. Skripto je bilo treba le dodati želenemu možicu in ta je skrbela za izris njegovega gibanja. Skripto smo prilagodili svojim potrebam, saj je bila napisana tako, da je delovala v povezavi z drugimi skriptami paketa Kinect with MS-SDK, ki so bile odgovorne za Kinectovo zaznavanje gibanja. Te odvisnosti smo odstranili, saj smo v svojem primeru imeli zaznavanje gibanja opravljeno na strežniku, in jo povezali s skripto, omenjeno v prejšnjem koraku, od katere je prejemala podatke gibanja. Nova skripta je naprej mapirala sklepe, ki jih zaznava Kinect, na kosti uporabljenega modela možica, nato pa odčitala začetne podatke o možičevem položaju in rotaciji. V funkciji Update, ki je med delovanjem aplikacije neprestano tekla, sta se izvedli dve stvari. Metoda MoveAvatar je glede na podatke o položaju telesa, ki smo jih dobili iz Kinecta, premikala možica na nov položaj. Druga stvar je bila transformacija kosti glede na rotacijske podatke sklepov. Zaradi tega je bilo potrebno, da smo v prejšnjih korakih iz pozicijskih podatkov o sklepih izračunali rotacijske matrike in na koncu dobili kvaternione. Gibanje človeka (z izjemo spreminjanja njegovega položaja) je bilo opravljeno izključno z uporabo rotacijskih podatkov.

4.3 Aplikacija Hoja nad prepadom



Slika 4.4: Hoja nad prepadom

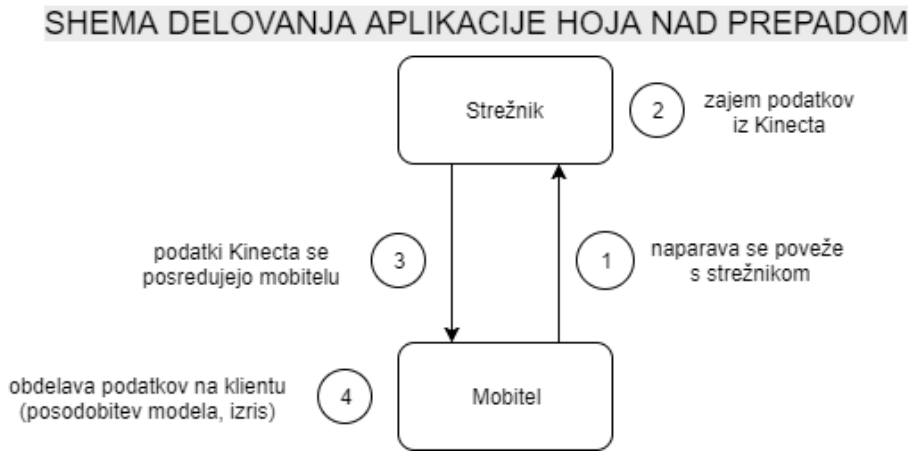
Ideja za aplikacijo Hoja nad prepadom je bila igra, v kateri igralec hodi po tanki palici, ki je postavljena visoko v zraku, pod njo pa je prepad. Igralec mora tako loviti ravnotežje in skrbeti, da ne pade z palice.



Slika 4.5: Igralčev pogled

Scena aplikacije je bila sestavljena iz naslednjih elementov:

- GameObject Terrain: S pomočjo orodja za dvigovanje in spuščanje reliefa smo ustvarili močno razgiban, hribovit svet.



Slika 4.6: Shema delovanja aplikacije Hoja nad prepadom

- GameObject Directional Light: Ta komponenta svetilnosti je zelo uporabna za ustvarjanje efektov sončne svetlobe [54]. V tem primeru aplikacije, ko igralec stoji precej visoko nad tlemi, si lahko med hojo po palici tudi ogleduje pokrajino okrog sebe. Zaradi tega je izbrana ravno ta svetloba, saj s svojimi efekti ustvari še lepši ambient.

- Prefab WaterProNighttime: S pomočjo prefaba, ki smo ga dobili iz uvožene Unityjeve zbirke Environment, smo ustvarili premikajoče jezero.

- GameObject Cube: Podlaga, po kateri se je igralec lahko premikal, je bila sestavljena iz dveh različnih kock, ki so vsebovale leseno teksturo ter Box Collider. Collider je komponenta, ki definira obliko objekta, ki mu pripada, namenjena pa je upravljanju fizičnih trkov. Ni pa obvezno, da se Collider popolnoma prilagaja obliki objekta, velikokrat je grobo približevanje bolj učinkovito za razvoj in igranje [7].

- UI GameObject Canvas: To je objekt, v katerega naj bi bili postavljeni vsi elementi UI aplikacije.. Naš je bil sestavljen iz več gradnikov. Vseboval je vnosno polje 4.7, ki se je pokazalo na začetku aplikacije,

v katero je uporabnik vpisal strežnikov IP-naslov, tekstovni element z besedilom 'Game Over!' in sliko črne barve. Zadnje dva elementa sta bila uporabljena za kreiranje animacije, ki se je izvedla ob koncu igre.

Slika 4.7: Začetno vnosno polje za vpis strežnikovega IP-naslova

- Prefab U-Character-REF: To je model možica, ki smo ga dobili iz paketa Kinect with MS-SDK. V zgornji del telesa, v višini oči, smo mu postavili kamero. Ta je bila usmerjena v isti smeri, kot je bil možicev obraz, tako da je igralec dobil občutek prvoosebnega pogleda. Modelu smo dodali tudi element RigidBody, katerega naloga je nadzorovanje položaja prek simulacij fizike. Gibanje predmeta, ki se mu dodeli komponenta RigidBody, bo pod nadzorom Unityjevega fizikalnega sistema. Nanj bo delovala težnost, če predmet vsebuje še komponento Collider, pa se bo ta lahko tudi odzival na trke z drugimi predmeti [43]. Dodan mu je bil tudi Collider, kar je omogočalo modelu, da je lahko hodil po podlagi, ki je tudi sama vsebovala Collider, ko pa je igralec zgubil Collider pod nogami, ga je težnost potegnila navzdol. Prefabu smo dodali tudi skripto GvrViewer, ki smo jo dobili v Googlovem VR SDK-ju za Unity [9]. Ta nam omogoča enostavno izbiranje med VR in navadnim načinom znotraj aplikacije.
- Skripta Reset: Ta je vsak trenutek preverjala igralčevo pozicijsko višino, in če je ta padla pod določeno mejo (kar je pomenilo padec), je sprožila

animacijo 'Game Over!' in čez nekaj sekund ponovno zagnala aplikacijo in začela novo igro.

4.4 Aplikacija Rokoborba

Rokoborba je aplikacija, namenjena dvema igralcema, ki se v boksarskem ringu pomerita s pestmi. Vsak igralec poskuša svojega nasprotnika s pestjo zadeti v glavo in mu tako zmanjšati življenjske točke. Nasprotnik se udarcu lahko izmakne ali pa se brani. Igre je konec, ko enemu izmed igralcev zmanjka življenjskih točk. Scena Rokoborbe je naslednja:

Slika 4.8: Igralčev pogled

- UI GameObject Canvas 1: Ta je vseboval tri barvne slike, dve tekstovni komponenti ter dva prikazovalnika življenjskih točk. Dve barvni sliki (temno rdeče in zelene barve) in obe tekstovni komponenti smo uporabili za kreiranje končne animacije. Zagon animacije je upravljala skripta, ki je v funkciji Update neprekinjeno preverjala, ali je kateri od igralcev izgubil vse življenjske točke. Ko se je to zgodilo, smo s pomočjo Animatorja sprožili stanje zmage ali poraza. V primeru zmage se je prikazala zelena animacija z napisom 'Victory!', v nasprotnem primeru pa je ekran postal temno rdeče barve z napisom 'Game Over!'. Preostalo barvno sliko (rdeče barve) smo uporabili za kratko animacijo, ki se je

predvajala, ko je prišlo do udarca v glavo. Prikazovalnika življenjskih točk sta bila sestavljena iz slike srca ter drsnika, v katerem je bilo vidno trenutno stanje igralčevih življenjskih točk.

- UI GameObject Canvas 2: Gradnik je vseboval eno ali dve vnosni polji, ki sta bili prikazani na začetku igre. V prvo vnosno polje je igralec vnesel IP-naslov računalnika, na katerega je bil povezan njegov Kinect. V drugo vnosno polje pa je vpisal IP-naslov svojega soigralca. Ko je igralec vpisal dva veljavna IP-naslova, sta se vnosni polji skrili.

- GameObject Point Light: Point Light je komponenta svetlobe, locirana na določeni točki, in oddaja svetlobo v vse smeri enako. Služila nam je za osvetlitev boksarskega ringa.

- Boksarski ring: Sestavili smo ga iz dveh različnih gradnikov, in sicer iz osmih kock (GameObject Cube) ter iz osmih valjev (GameObject Cylinder). Gradnikom smo dodelili tudi različne teksture lesa, ki smo jih uvozili iz Unityjeve trgovine [2].

Slika 4.9: Rokoborba

- Prefab Igralec Dimples : Dimples je model možica, ki smo ga uvozili iz knjižnice Dimples - Boxing Character model. Tudi v ta model smo kot v prejšnjem primeru pripeli kamero. Kot smo že prej omenili, je cilj

igre s pestjo zadeti nasprotnika v glavo, in zato smo morali določiti sistem za zaznavanje udarcev. Model je vseboval tri komponente Sphere Colliderjev, in sicer enega v predelu glave, druga dva pa v gradnikih leve in desne roke. Ročna Sphere Colliderja sta bila namenjena napadu ter obrambi, naloga naglavnega Sphere Colliderja pa je bila, da je zaznal, ko so se drugi Colliderji zadeli ob njega. To smo omogočili s tem, da smo naglavni komponenti obkljukali možnost 'is Trigger'. Ko je drug Collider zadel ob njega, je bila klicana funkcija OnTriggerEnter, funkcija OnTriggerExit pa se je sprožila, ko se je stik med Colliderjema prekinil. Gradnikom ročnih Colliderjev smo dodelili značke, da je naglavni Sphere Collider lahko ločil med lastnimi in nasprotnikovimi pestmi. Trk z eno izmed nasprotnikovih rok je pomenil prejet udarec, hkraten trk z obema lastnima rokama pa obrambo.

- Prefab Nasprotnik Dimples: Ta model je predstavljal premikanje nasprotnega igralca. Tudi ta je vseboval enak sistem Sphere Colliderjev kot prejšni model. S pestmi ga je bilo treba zadeti v glavo in se izmikati njegovim udarcem. Igralec ga je porazil tako, da mu je z udarci zmanjšal število življenjskih točk na nič.

- Skripti UserAttacked in EnemyAttacked: Skripti sta skrbeli za določanje akcij v igri: udarec ali obramba. Vsebovali sta funkciji OnTriggerEnter in OnTriggerExit, ki sta bili klicani ob različnih trkih teles. S pomočjo različnih oznak (vsak igralec je imel svojo oznako), se je določilo, ali je trk pomenil udarec ali obrambo. Ta informacija se je nato posredovala skripti PlayerHealth.

- Skripta PlayerHealth: Skripta je vsak trenutek preverjala, ali je prišlo do udarca, in v tem primeru ustreznemu igralcu zmanjšala življenjske točke in informacijo posredovala drugi napravi. Upravljala je tudi z zvočnimi efekti ter ob udarcu sprožila krajšo animacijo.

- Skripta newGame: Ta je vsak trenutek preverjala, ali je kateremu od igralcev zmanjkalo življenjskih točk, in glede na to, ali je igralec zgubil

ali zmagal, sprožila pravilno animacijo. Ko se je animacija zaključila, je ponovno zagnala aplikacijo in začela novo igro.

Aplikacija pa ni bila popolnoma enaka na obeh računalnikih. Na namiznem računalniku je bila postavljena gostujoča aplikacija, na prenosnem pa aplikacija, ki se je povezala na gostujočo.

Prva razlika je bila povezljivost. Glavni aplikaciji se ob zagonu prikaže poleg vnosnega polja, namenjenega IP-naslovu svojega strežnika, tudi vnosno polje, namenjeno IP-naslovu svojega soigralca. Ko uporabnik vpiše veljavni IP-naslov, se s pomočjo povezave UDP na dani naslov pošlje en bajt veliko sporočilo. Aplikacija, ki ni gostujoča, ob svojem zagonu začne nemudoma poslušati ali se želi kdo povezati z njo, gostujoča aplikacija ima nato natanko 15 sekund, da drugi pošlje sporočilo in ji tako omogoča nadaljnje delovanje (ta je med poslušanjem na čakanju). Če se v tem času aplikaciji ne povežeta, igra ne bo pravilno delovala in je treba ponovno zagnati obe aplikaciji.

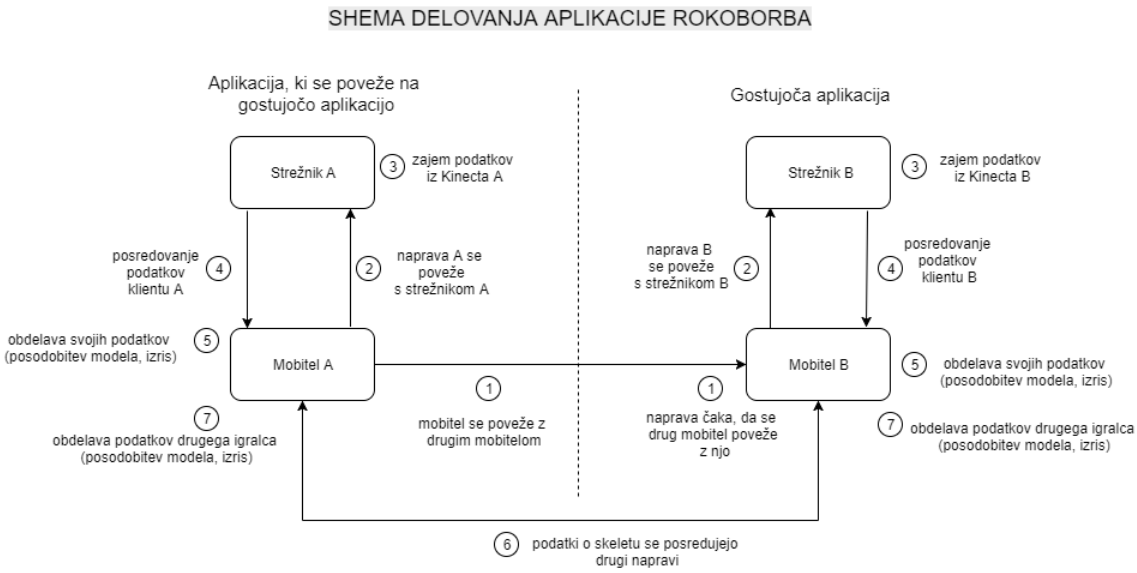
Druga razlika pa je proces določanja napadov in obrambe. Zaradi možnih razlik v delovanju aplikacij smo se odločili, da bo samo ena aplikacija skrbela za določanje napadov in obrambe. Zato poteka zaznavanje udarcev in drugih akcij samo na gostujoči aplikaciji, ta pa drugi aplikaciji prek povezave UDP posreduje informacije o akciji, ki se je zgodila.

4.5 Optimizacija in izboljšave

Možnosti za izboljšavo aplikacije je veliko. Najprej bomo predstavili izboljšave, ki bi bile primerne za obe aplikaciji, nato pa bomo za vsako aplikacijo posebej našteli in opisali izboljšave, ki bi popestrile in izboljšale njihovo igralniško izkušnjo.

4.5.1 Izboljšave aplikacij

Prva izboljšava bi bila zaznavanje posameznih prstov. Ko smo uporabljali Kinect for XBox 360, je bila ta izboljšava nemogoča, novejši Kinect pa nam



Slika 4.10: Shema delovanja aplikacije Rokoborba

je to možnost omogočal in bi tako lahko še izboljšali uporabnikov prikaz v navidezni resničnosti. Druga izboljšava je izrisovanje polja, v katerem Kinect še zazna igralca. S pomočjo tega bi uporabnik vedel, kje se lahko premika ter koliko se lahko približa oziroma oddalji od senzorja Kinect. Igra bi tako potekala bolj tekoče, saj bi Kinect v vsakem trenutku zaznavanja odčital podatke vseh sklepov. Možna izboljšava bi bila tudi dodajanje VR-komponent v aplikacijo. Trenutno je za delovanje aplikacije na mobilni napravi potrebno ročno upravljanje. S pomočjo VR-komponent bi upravljanje aplikacije lahko potekalo prek naglavnega prikazovalnika.

4.5.2 Izboljšave aplikacije Hoja nad prepadom

Možna izboljšava aplikacije bi bila bolje dodelana okolica igre. Prostor, v katerega je postavljen igralec, bi lahko nadgradili z bolj atraktivnimi elementi, dodali bi mu lahko modele živali in drugih samodejno premikajočih se objektov, kar bi naredilo igralniško izkušnjo toliko bolj realistično. Dobra nadgradnja aplikacije bi bila možnost izbiranja med različnimi okolji, v ka-

tere je igralec postavljen. Ustvariti bi morali več različnih scenarijev, igralec pa bi ob zagonu aplikacije izbral, v katerem izmed njih želi preizkusiti svoje spretnosti hoje po palici. Naslednja izboljšava bi bila boljši sistem za simulacijo hoje po palici. V trenutni različici naše aplikacije ima model igralca v predelu nog en sam Box Collider, ki je odgovoren za določanje, ali igralec stoji na palici ali ne. Tak pristop ne zagotavlja najboljše natančnosti. Izboljšali bi ga z dodajanjem še enega Box Colliderja, tako da bi vsak Collider predstavljal svojo nogo. Igralec bi tako imel več svobode pri premikanju nog.

4.5.3 Izboljšave aplikacije Rokoborbe

Prva izboljšava bi bila optimizacija strukture aplikacije. Kot smo že povedali, je aplikacija namenjena dvema igralcema, ki pa imata na svojih računalnikih malce drugačni verziji aplikacije. Strukturo bi spremenili tako, da bi na imela igralca isto aplikacijo, ki bi obema omogočala izbiranje soigralca in vodenje igre. Naslednja izboljšava bi bila na področju napadanja. Spremenili bi način bojevanja. Poleg udarcev z roko bi uvedli tudi napad z nogo. Tarča nasprotnika ne bi bila več samo glava, ampak celotno telo, ki bi ga razdelili na več različnih vrednostnih območij. Tako bi udarec v glavo igralcu zbil več življenjskih točk kot pa udarec v trebuh. Moč udarca pa bi nadgradili tudi z njegovo hitrostjo. Bolj ko je udarec hiter, več življenjskih točk bi zbil. Aplikacijo bi lahko izboljšali tudi tako, da bi dodali prikazovalnik trenutnega rezultata tekem. Aplikacija bi hranila podatek o tem, koliko zmag in porazov ima vsak od igralcev. Možna izboljšava bi bila tudi možnost igranja proti igralcu, ki nima Kinecta. Ta bi svojega boksarja upravljal s pomočjo miške in tipkovnice. Za zdaj se na skupno igro lahko povežeta le igralca, ki sta povezana v isto omrežje. Aplikacijo bi lahko nadgradili tako, da bi omogočala igranje prek interneta. Naslednja izboljšava bi bila izbira modela boksarja. V Unityjevi trgovini je veliko modelov, ki bi jih lahko uvozili v svojo aplikacijo, igralca pa bi ob zagonu igre lahko izbirala, s katerim izmed ponujenih modelov bi želela igrati.

4.6 Testiranje aplikacij

Razviti aplikaciji smo na koncu testirali in ovrednotili.

Vse performančne teste prve aplikacije smo izvajali na osebnem računalniku (glej oddelek 4.1), na Kinectu for XBox One ter na mobilnem telefonu Moto G5 Plus z naslednjimi specifikacijami:

- procesor: Osemjedrni 2.0 GHz,
- chipset: Qualcomm MSM8953 Snapdragon 625,
- pomnilnik: 3 GB,
- operacijski sistem: Android 7.0 Nougat,
- grafična procesna enota: Adreno 506.

Spremljali smo povratni čas pri prenosu podatkov med napravama ter zamik med zajemom podatkov o skeletu s senzorjem Kinect ter izrisom teh v aplikaciji. Problem, ki se nam je pojavljal pri merjenju zamikov med napravami, je neusklajenost ur naprav (vsaj ne do milisekundnih ali višjih natančnosti). Zaradi tega so meritve potekale le na strani strežnika, kjer smo s pomočjo povratnih sporočil in z izračunom povratnega časa lahko določili še ostale zamike.

Za merjenje povratnega časa pri prenosu podatkov med napravama smo aplikacijo prilagodili tako, da je ta ob prejetem sporočilu strežnika to isto sporočilo takoj posredovala nazaj. Na strežniku smo s pomočjo razreda Stopwatch merili čas, ki je bil potreben, da se je poslano sporočilo vrnilo nazaj. Po več meritvah smo zabeležili, da je povprečni povratni čas pri prenosu podatkov med napravama približno enak 9,02 ms. S pomočjo tega podatka si lahko ustvarimo približno oceno, da je zamik pri prenosu med napravama približno enak 4,51 ms. Med samim ocenjevanjem povratnega časa smo prišli do ugotovitve, da je ta odvisen tudi od števila sklepov, ki jih zazna senzor Kinect, saj to vpliva na velikost sporočila. Boljše ko je zaznan človeški skelet, več informacij o sklepih bomo posredovali naprej, večje bo sporočilo in

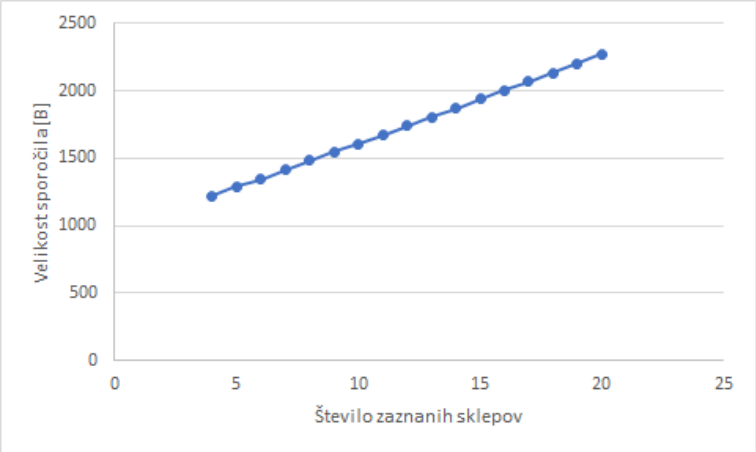
posledično večji povratni čas. Seveda pa se moramo zavedati, da je povratni čas močno odvisen tudi od omrežja, v katerem smo.

Število sklepov	Velikost [B]	Povratni čas [ms]
4	1220	4,9
7	1412	5,1
10	1607	6,5
13	1802	7,5
l7	2066	8,8
20	2274	9,3

Tabela 4.1: Rezultati povratnega časa glede na število zaznanih sklepov

Slika 4.11: Graf povratnega časa v odvisnosti od števila zaznanih sklepov

Merjenje zamika od zajetja Kinectovih podatkov do izrisa v aplikaciji pa je potekalo v dveh korakih. V prvem koraku smo morali določiti povprečni čas, ki je potreben od zajetja podatkov do izrisa in povratne informacije na strežnik. V trenutku, ko je na strežniku Kinect zaznal človeško postavo, smo začeli meriti čas. Strežnik je nato podatke poslal aplikaciji in čakal na odgovor. Ko je aplikacija končala izris, je bilo sporočilo posredovano nazaj na strežnik, kjer se je zabeležil porabljeni čas. Po več meritvah smo zabeležili



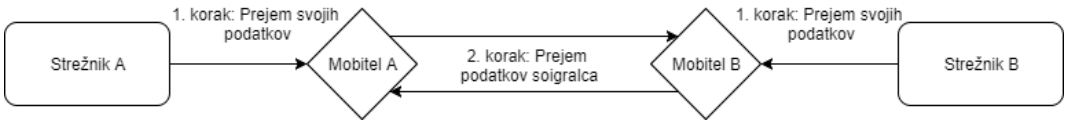
Slika 4.12: Graf velikosti sporočila v odvisnosti od števila zaznanih sklepov

povprečni čas 56,25 ms. V drugem koraku smo od dobljenega povprečja odšteli zamik prenosa podatkov med napravama (4,51 ms) in tako smo dobili iskani zamik, ki je znašal 51,74 ms.

Performančne teste druge aplikacije smo izvajali na osebnem in na prenosnem računalniku, katerih specifikacije smo navedli že na začetku odlomka. Uporabljali smo oba Kinecta ter dve mobilni napravi: Moto G5 Plus ter Elephone P9000 z naslednjimi specifikacijami:

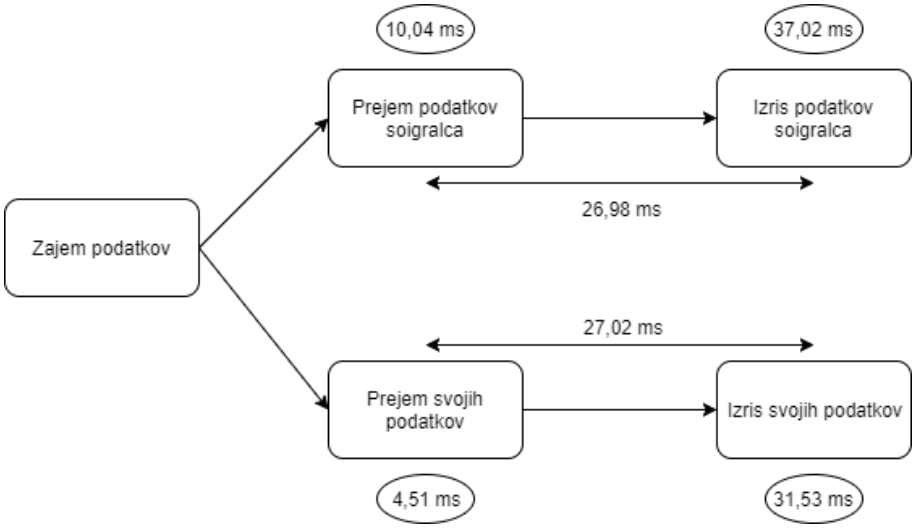
- procesor: Osemjedrni 2.0 GHz,
- chipset: Helio P10 MTK6755,
- pomnilnik: 4 GB,
- operacijski sistem: Android 6.0 Marshmallow,
- grafična procesna enota: ARM Mali-T860.

Najprej smo spremljali povratni čas sporočil. Meritve, ki smo jih izvajali med dvema napravama, so nam vračale iste rezultate kot prejšnja aplikacija (9,02 ms). Nato smo izmerili povratni čas pri treh napravah: računalniku in dveh mobilnih napravah. Izmerili smo povratni čas sporočila, ki je bilo najprej s strežnika poslano prvi mobilni napravi. Ta je sporočilo posredovala drugi mobilni napravi, ki je sporočilo spet poslala strežniku. Po več meritvah smo prišli do rezultata 15,05 ms. Če predpostavljamo, da je za vsak korak sporočilo potrebovalo približno enako dolgo, ugotovimo, da je zamik med napravami bil 5,02 ms, kar je dober rezultat, saj se od prvega približka zamika razlikuje le za 0,51 ms. Od tukaj sledi, da potrebujemo 10,04 ms, preden dobimo podatke o soigralčevem skeletu, saj sta za to potrebna dva koraka.



Slika 4.13: Diagram prenašanja podatkov po sistemu

Nadalje smo izmerili zamik med Kinectovim zajemom podatkov o igralcu ter izrisom teh podatkov na igralčevi mobilni napravi. Zamik smo merili na mobilnemu telefonu Moto G5 Plus, saj je ta boljši od druge mobilne naprave (Elephone P9000) in nam je vračal natančnejše rezultate. Povprečni zamik od zajema podatkov do izrisa je bil 36,04 ms. Če temu času odstranimo zamik med dvema napravama, ki znaša 4,51 ms, dobimo končni zamik 31,53 ms. Merili smo tudi čas, ki je potreben, da naprava izriše podatke svojega soigralca. Po več poskusih smo prišli do rezultata 42,04 ms. Temu času smo odšteli zamik povratne informacije (5,02 ms) in prišli do končnega zamika, ki znaša 37,02 ms.



Slika 4.14: Časovni diagram poteka aplikacije

Iz rezultatov in diagrama lahko sklepamo, da se največji zamik podatkov ustvari pri obdelovanju in izrisovanju podatkov. Zanimivo je tudi dejstvo, da naprava potrebuje približno enak čas za izris igralcev. Izris igralcev pa se na mobilni napravi ne začne istočasno, saj dobi ta podatke o igralcih ob različnih časih. Naprava dobi podatke o svojem igralcu v enem samem koraku (strežnik A $\implies$ mobilna naprava A), podatke o njegovem soigralcu pa v dveh (strežnik B $\implies$ mobilna naprava B $\implies$ mobilna naprava A). Zaradi tega se izris svojega igralca začne malce prej kot pa izris soigralca.

V nasprotju s pričakovanji je zamik med zajemom podatkov in izrisom manjši v aplikaciji Rokoborba, čeprav ta izrisuje podatke dveh oseb. To je verjetno posledica bolj kompleksnega okolja navidezne resničnosti v prvi aplikaciji. V aplikaciji Rokoborba je prostor zelo enostaven, sestavljen le iz osnovnih gradnikov, pri Hoji nad prepadom pa je okolica sestavljena iz kompleksnejših objektov, kot sta premikajoče se morje in hriboviti teren. Mogoč dejavnik je lahko tudi to, koliko okolice zajema igralčev pogled. V prvi aplikaciji je igralec postavljen precej visoko, njegov pogled je zaradi tega zelo širok in zajema velik del okolice, ki jo mora naprava vsak trenutek izrisovati. Pri aplikaciji Rokoborba pa je igralec postavljen na tla in vidi le nasprotnega boksarja in del ringa, kar pa GPU-ju ne predstavlja večjega napora pri izrisovanju.

Poglavje 5

Zaključek

Uspelo nam je vzpostaviti sistem, ki združuje Kinectovo zaznavanje gibanja z navidezno resničnostjo. Res je, da je na trgu že veliko naprav (HTC Vive, VicoVR, Virtuix Omni, Oculus Rift), ki ponujajo izkušnjo navidezne resničnost, povezane z zaznavanjem gibanja, ampak so te precej drage. Nam je s spojitvijo Kinecta in Googlovega Cardboarda uspelo vzpostaviti dokaj poceni sistem, ki je enostaven za uporabo. Treba je le priklopiti senzor Kinect v računalnik, natakniti si naglavni prikazovalnik in zabava se lahko začne.

S pomočjo mnogo knjižnic, SDK-jev, Unityjevih paketov in naše kode smo sestavili model, ki se lahko uporabi za nadaljnje razvijanje aplikacij, namenjenih Kinectu in VR. Mi smo ustvarili dve taki aplikaciji: Hojo nad prepadom ter Rokoborbo. Prva je namenjena enemu igralcu, ki s pomočjo Kinecta v navidezni resničnosti poskuša ostati na palici in ne pasti v prepad. V drugi aplikaciji, ki je namenjena dvema igralcema, pa se ta dva pomerita v igri rokoborbe.

V fazi testiranja smo ugotovili, da sta naši aplikaciji dobro implementirani in optimizirani, saj je zamik med zajemom podatkov s senzorjem Kinect in izrisom v aplikaciji pri obeh aplikacijah minimalen.

S tema aplikacijama smo želeli predstaviti različne možnosti uporabe dane tehnologije. Seveda pa je to šele začetek. Po bolj obširnem pregledu področij smo ugotovili, da postaja uporaba opisanih tehnologij popularnejša in bolj

in bolj prisotna na mnogo področjih. Namen uporabe Kinecta in navidezne resničnosti se je od svojih začetkov do danes zelo razširil, razvoja pa še zdaleč ni konec. V prihodnosti bodo po našem mnenju ti dve tehnologiji svoj položaj še utrdili, njihovo uporabnost pa bodo odkrili še na drugih področjih.

Literatura

[1] 10 ways virtual reality is revolutionizing medicine and healthcare. Dosegljivo: https://www.techrepublic.com/article/10-ways-virtual-reality-is-revolutionizing-medicine-and-healthcare/. [Dostopano: 25. 7. 2017].

[2] 15 original wood texture. Dosegljivo: https://www.assetstore.unity3d.com/en/#!/content/71286. [Dostopano: 21. 8. 2017].

[3] 3 ways med students can use virtual reality. Dosegljivo: https://edtechmagazine.com/higher/article/2016/08/three-ways-med-students-can-use-virtual-reality. [Dostopano: 23. 7. 2017].

[4] 3d serious games for parkinson's disease management. Dosegljivo: https://http://hci.si/2016/11/03/3d-serious-games-parkinsons-disease-management/. [Dostopano: 25. 8. 2017].

[5] The ars review: Oculus rift expands pc gaming past the monitor's edge. Dosegljivo: https://arstechnica.com/gaming/2016/03/the-ars-review-oculus-rift-expands-pc-gaming-past-the-monitors-edge/. [Dostopano: 18. 8. 2017].

[6] Changing the face of healthcare with vr. Dosegljivo: https://filmora.wondershare.com/virtual-reality/healthcare-with-virtual-reality.html. [Dostopano: 23. 7. 2017].

[7] Colliders. Dosegljivo: https://docs.unity3d.com/Manual/CollidersOverview.html. [Dostopano: 21. 8. 2017].

[8] Common language runtime (clr). Dosegljivo: https://docs.microsoft.com/en-us/dotnet/standard/clr. [Dostopano: 15. 8. 2017].

[9] Downloads and samples. Dosegljivo: https://developers.google.com/vr/unity/download. [Dostopano: 21. 8. 2017].

[10] Gear vr. Dosegljivo: http://www.samsung.com/global/galaxy/gear-vr/. [Dostopano: 18. 8. 2017].

[11] Gear vr with controller. Dosegljivo: http://s7d2.scene7.com/is/image/SamsungUS/2_74747_GearVR_007_Set1_Black_517325990_517325991_303957788?$product-details-jpg$. [Dostopano: 30. 7. 2017].

[12] Google cardboard is vr's gateway drug. Dosegljivo: https://www.wired.com/2015/05/try-google-cardboard/. [Dostopano: 16. 8. 2017].

[13] History of virtual reality. Dosegljivo: https://www.vrs.org.uk/virtual-reality/history.html. [Dostopano: 1. 6. 2017].

[14] How does the kinect 2 compare to the kinect 1. Dosegljivo: http://zugara.com/how-does-the-kinect-2-compare-to-the-kinect-1. [Dostopano: 16. 8. 2017].

[15] How microsoft kinect works. Dosegljivo: http://electronics.howstuffworks.com/microsoft-kinect2.htm. [Dostopano: 16. 8. 2017].

[16] Htc vive. Dosegljivo: https://www.techspot.com/products/audio-video/htc-vive.131920/. [Dostopano: 19. 8. 2017].

[17] Htc vive: Everything you need to know about the steamvr headset. Dosegljivo: https://www.wareable.com/vr/htc-vive-vr-headset-release-date-price-specs-7929. [Dostopano: 18. 8. 2017].

[18] Htc vive vr virtual reality headset gaming system. Dosegljivo: https://www.scan.co.uk/images/products/2689649-a.jpg. [Dostopano: 30. 7. 2017].

[19] Jedibot. Dosegljivo: http://www.scififx.com/wp-content/uploads/2011/07/JediBot.jpg. [Dostopano: 2. 7. 2017].

[20] Json.net. Dosegljivo: https://www.newtonsoft.com/json. [Dostopano: 20. 8. 2017].

[21] Kinect. Dosegljivo: https://en.wikipedia.org/wiki/Kinect. [Dostopano: 17. 8. 2017].

[22] Kinect for windows sdk 2.0. Dosegljivo: https://www.microsoft.com/en-us/download/details.aspx?id=44561. [Dostopano: 20. 8. 2017].

[23] Kinect for windows sdk v1.8. Dosegljivo: https://www.microsoft.com/en-us/download/details.aspx?id=40278. [Dostopano: 20. 8. 2017].

[24] Kinect for xbox one. Dosegljivo: http://compass.xbox.com/assets/89/a6/89a6cdd2-28f5-4b62-a4cd-88d910954d7e.jpg?n=X1-Kinect-Sensor_Feature-1400_Voice-Commands_800x450.jpg. [Dostopano: 21. 7. 2017].

[25] Kinect to vr. Dosegljivo: https://channel9.msdn.com/coding4fun/kinect/Kinect-to-VR. [Dostopano: 2. 6. 2017].

[26] Kinect v1 and kinect v2 fields of view compared. Dosegljivo: http://smeenk.com/kinect-field-of-view-comparison/. [Dostopano: 18. 8. 2017].

[27] Kinect with ms-sdk. Dosegljivo: https://www.assetstore.unity3d.com/en/#!/content/7747. [Dostopano: 21. 8. 2017].

[28] Lightsaber + kinect + robotic arm = jedibot. Dosegljivo: https://www.extremetech.com/extreme/90204-lightsaber-kinect-robotic-arm-jedibot. [Dostopano: 2. 6. 2017].

[29] Microsoft visual studio. Dosegljivo: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio. [Dostopano: 15. 7. 2017].

[30] Microsoft xbox 360 kinect review. Dosegljivo: https://cnet4.cbsistatic.com/img/UQWXVrGnqRBLfBQKhsW4QLHBlqM=/830x467/2010/11/05/220be19f-cbf2-11e2-9a4a-0291187b029a/orig-1200x900_1.jpg. [Dostopano: 21. 7. 2017].

[31] Motion detection. Dosegljivo: https://en.wikipedia.org/wiki/Motion_detection. [Dostopano: 29. 5. 2017].

[32] Net framework. Dosegljivo: https://upload.wikimedia.org/wikipedia/commons/thumb/d/d3/DotNet.svg/300px-DotNet.svg.png. [Dostopano: 2. 8. 2017].

[33] .net framework. Dosegljivo: https://en.wikipedia.org/wiki/.NET_Framework. [Dostopano: 15. 7. 2017].

[34] .net framework class library overview. Dosegljivo: https://docs.microsoft.com/en-us/dotnet/standard/class-library-overview. [Dostopano: 15. 8. 2017].

[35] Oculus rift. Dosegljivo: https://en.wikipedia.org/wiki/Oculus_Rift. [Dostopano: 18. 8. 2017].

[36] Oculus rift. Dosegljivo: https://www.techspot.com/products/audio-video/oculus-rift.131668/. [Dostopano: 19. 8. 2017].

[37] Oculus rift vr headset with touch vr controllers. Dosegljivo: https://multimedia.bbycastatic.ca/multimedia/products/500x500/b00/b0008/b0008225.jpg. [Dostopano: 30. 7. 2017].

[38] Playstation vr. Dosegljivo: https://media.playstation.com/is/image/SCEA/vr-refresh-tech-set?$TwoColumn_Image$. [Dostopano: 25. 7. 2017].

[39] Playstation vr. Dosegljivo: https://en.wikipedia.org/wiki/PlayStation_VR. [Dostopano: 18. 8. 2017].

[40] Playstation vr. Dosegljivo: https://www.techspot.com/products/audio-video/playstation-vr.152662/. [Dostopano: 19. 8. 2017].

[41] Playstation vr review. Dosegljivo: http://www.techradar.com/reviews/gaming/playstation-vr-1235379/review. [Dostopano: 18. 8. 2017].

[42] The pros and cons of google cardboard. Dosegljivo: https://www.technavio.com/blog/the-pros-and-cons-of-google-cardboard. [Dostopano: 19. 8. 2017].

[43] Rigidbody. Dosegljivo: https://docs.unity3d.com/ScriptReference/Rigidbody.html. [Dostopano: 21. 8. 2017].

[44] Samsung gear vr. Dosegljivo: https://www.techspot.com/products/audio-video/samsung-gear-vr.131921/. [Dostopano: 19. 8. 2017].

[45] Samsung gear vr consumer edition goes on sale in the us. Dosegljivo: http://www.trustedreviews.com/news/samsung-gear-vr-consumer-edition-goes-on-sale-in-the-us-2929981. [Dostopano: 18. 8. 2017].

[46] Sensorama. Dosegljivo: https://en.wikipedia.org/wiki/Sensorama. [Dostopano: 25. 7. 2017].

[47] Six virtual reality apps you can use with google cardboard. Dosegljivo: http://cdn2.hubspot.net/hubfs/307727/Stock_Photos/Cardboard.jpg. [Dostopano: 21. 7. 2017].

[48] Snow world. Dosegljivo: https://i2.wp.com/sensics.com/wp-content/uploads/2016/01/Firsthand_04bis1.jpg?resize=300%2C169. [Dostopano: 22. 7. 2017].

[49] Sony announces project morpheus, a virtual reality headset coming to playstation 4. Dosegljivo: https://www.polygon.com/2014/3/18/5524058/playstation-vr-ps4-virtual-reality. [Dostopano: 18. 8. 2017].

[50] Stanford's 'jedibot' tries to kill you with a foam sword. Dosegljivo: http://spectrum.ieee.org/automaton/robotics/diy/stanford-robots-flip-burgers-play-jedi-make-your-life-complete. [Dostopano: 2. 6. 2017].

[51] Sword of damocles vr. Dosegljivo: https://roadtovrlive-5ea0.kxcdn.com/wp-content/uploads/2016/05/ultimate-display.jpg. [Dostopano: 25. 7. 2017].

[52] Tcp vs. udp. Dosegljivo: http://www.diffen.com/difference/TCP_vs_UDP. [Dostopano: 15. 8. 2017].

[53] Tracked skeleton joints. Dosegljivo: https://www.researchgate.net/profile/Marija_Mihova/publication/267267474/figure/fig1/AS:295630367936530@1447495180618/Figure-1-Tracked-skeleton-joints-of-the-user%27s-body-Kinect-is-also-appropriate-for.png. [Dostopano: 15. 8. 2017].

[54] Types of light. Dosegljivo: https://docs.unity3d.com/Manual/Lighting.html. [Dostopano: 21. 8. 2017].

[55] Understanding how depth sensing cameras work. Dosegljivo: http://www.vivekc.com/understanding-how-depth-sensing-cameras-work/. [Dostopano: 17. 8. 2017].

[56] Unity - store. Dosegljivo: https://store.unity.com/?_ga=2.79126996.1940323939.1503147417-1632842766.1498765369. [Dostopano: 20. 8. 2017].

[57] Unity (game engine). Dosegljivo: https://en.wikipedia.org/wiki/Unity_(game_engine). [Dostopano: 15. 7. 2017].

[58] Valve's vr headset is called the vive and it's made by htc. Dosegljivo: https://www.theverge.com/2015/3/1/8127445/htc-vive-valve-vr-headset. [Dostopano: 18. 8. 2017].

[59] Virtual boy. Dosegljivo: https://upload.wikimedia.org/wikipedia/commons/thumb/4/44/Virtual-Boy-Set.jpg/1200px-Virtual-Boy-Set.jpg. [Dostopano: 25. 7. 2017].

[60] Virtual reality. Dosegljivo: https://en.wikipedia.org/wiki/Virtual_reality. [Dostopano: 1. 6. 2017].

[61] Virtual reality game helps patients with ms. Dosegljivo: https://www.ksat.com/news/virtual-reality-game-helps-patients-with-ms. [Dostopano: 12. 7. 2017].

[62] Virtual reality in stroke rehabilitation. Dosegljivo: https://nydnrehab.com/wp-content/uploads/2014/02/2troke-Rehabilitation-Center-NYC1.jpg. [Dostopano: 30. 8. 2017].

[63] Virtual reality in stroke rehabilitation. Dosegljivo: https://nydnrehab.com/treatment-methods/neurorehab/virtual-reality-in-stroke-rehabilitation/. [Dostopano: 25. 7. 2017].

[64] Virtual reality pain reduction. Dosegljivo: http://www.vrpain.com/. [Dostopano: 26. 7. 2017].

[65] Virtual reality sharks nab professor 240,000 dollars grant. Dosegljivo: http://tpr.org/post/virtual-reality-sharks-nab-professor-240000-grant#stream/0. [Dostopano: 12. 7. 2017].

[66] Visual studio downloads. Dosegljivo: https://www.visualstudio.com/downloads/. [Dostopano: 20. 8. 2017].

[67] What is virtual reality? Dosegljivo: https://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html. [Dostopano: 29. 5. 2017].

[68] Franc Solina in Blaž Meden. Light fountain – a virtually enhanced stone sculpture. *Digital Creativity*, 28(2):89–102, 2017.

[69] Hossein Mousavi Hondori in Maryam Khademi. A review on technical and clinical impact of microsoft kinect on physical therapy and rehabilitation,. *Journal of Medical Engineering*, 2014:16, 2014.

[70] Kate Laver, Stacey George, Susie Thomas, Judith E. Deutsch, and Maria Crotty. Virtual reality for stroke rehabilitation. *Stroke*, 43(2):e20–e21, 2012.

[71] Oliver Wasenmüller and Didier Stricker. Comparison of kinect v1 and v2 depth images in terms of accuracy and precision. In *Asian Conference on Computer Vision Workshop. Asian Conference on Computer Vision Workshop (ACCV workshop-16), Taipeh, Taiwan, Province of China*. Springer, 2016.

[72] Brian M. Williamson, Dr. Joseph J. Laviola, Tim Roberts, and Pat Garrity. Interservice/industry training, simulation, and education conference (i/itsec) 2012 multi-kinect tracking for dismounted soldier training.

[73] Z. Zhang. Microsoft kinect sensor and its effect. *IEEE MultiMedia*, 19(2):4–10, Feb 2012.